FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Compressão e Encriptação de Dados Aplicado a Etiquetas de Tecnologia Near-Field Communication

Abel Almeida Maio

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Doutor Rui Rodrigues (Professor Auxiliar Convidado)

17 de Junho de 2013

Compressão e Encriptação de Dados Aplicado a Etiquetas de Tecnologia Near-Field Communication

Abel Almeida Maio

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: Doutor Pedro Souto (Professor Auxiliar)

Vogal Externo: Doutor André Zúquete (Professor Auxiliar)

Orientador: Doutor Rui Rodrigues (Professor Auxiliar Convidado)

---

18 de Julho de 2013

Resumo

Nos dias de hoje, a tecnologia está a avançar a um ritmo cada vez mais acelerado, aparecendo melhorias às tecnologias existentes, assim como novas tecnologias que abrem todo um novo conjunto de possibilidades.

Nos últimos anos tem aparecido e ganho mais popularidade uma nova tecnologia denominada *Near-Field Comunication* ou *NFC*, que permite comunicação entre dois pontos, com a particularidade de ser necessário esses dois pontos estarem extremamente próximos para a comunicação ocorrer. Para além de possibilitar a comunicação entre dois pontos, esta tecnologia permite também a existencia de etiquetas especiais que podem guardar pequenas quantidades de informação.

Contudo, esta nova tecnologia não tem mecanismos de proteção de leitura dessas etiquetas, permitindo que quando pessoa com um leitor suficientemente perto consiga recolher a informação que se encontra armazenada. De forma a proteger essa informação é necessário que a aplicação usada para interagir com a etiqueta tenha implementado mecanismos de proteção dessa mesma informação.

A questão que se levanta e que se pretende responder com este trabalho é, devido às limitações de tamanho das etiquetas e de processamento dos actuais dispositivos móveis, existirá alguma combinação de algoritmos que se mostre eficiente o suficiente para desempenhar essa tarefa, garantindo o melhor aprovietamento dos recursos e assegurando uma boa proteção da informação guardada.

Abstract

Nowadays technology is progressing at a faster pace every day, showing improvements to existing technologies as well as new technologies that are emerging, opening up a whole new set of possibilities.

In recent years, a new technology has appeared and it's gaining more and more popularity, called Near-Field Communication or NFC. This technology allows communication between two points, with the particularity of being required that this two points are close enought for the communication to occur. In addition, is possible to store different kinds of information in some special NFC enabled tags.

However, this new technology has no mechanisms for protecting the readability of this tags, allowing any person that is close enought to read the stored information. In order to protect the information, the application used to interact with the tag must implement itself the mechanisms for protecting the information.

The question that arises and that we intend to answer with this work is, due to size limitations of the tags and the limited resources of mobile devices, is there any combination of methods and algorithms that proves efficient enought to protect the data having in attention the limited nature of the resources.

Agradecimentos

Este trabalho simboliza uma fase muito importante para todos os estudantes. É uma fase de mudança, o final de uma época e o início de uma nova era. Todas as pessoas que nos influenciam nestes momentos são aqueles que nos apoiam e acreditam em nós e isso é mais valioso que ouro.

Gostaria de agradecer ao meu orientador, Professor Rui Rodrigues, por todas as sugestões e apoio e por não só ter acreditado em mim ao início desta fase, mas também por me ter apoiado até ao final.

Quero agradecer também a todos os meus amigos, tanto aqueles de longa data, como aqueles que fiz no meu percurso académico, amigos que provaram que o são realmente ao longo dos tempos, por me terem dado todo o tipo de apoio emocional e psicológico, ajudando-me a ultrapassar momentos difíceis que se atravessam na vida e servindo como motivação para continuar a lutar.

Finalmente, um agradecimento muito especial à minha família, que sempre esteve ao meu lado, dando-me todo o apoio a todos os níveis, sendo um dos verdadeiros pilares de suporte para todos os desafios que tenho ultrapassado ao longo da minha vida.

Abel Maio

"Anyone who has never made a mistake has never tried anything new."

Albert Einstein

Conteúdo

CONTEÚDO

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	Application Programming Interface
AES	Advanced Encryption Standard
DES	Data Encryption Standard
DESede	Triple Data Encryption Standard
KB	Kilo Byte
NFC	Near Field Communication
RFID	Radio-frequency Identification
SDK	Software Development Kit

Capítulo 1

Introdução

Nos dias de hoje, cada vez mais dispositivos móveis estão a ser usados como verdadeiros assistentes para a nossa vida pessoal e profissional. Com a evolução tecnológica é cada vez maior a quantidade de informação que é guardada nestes dispositivos, são criados novos modelos com mais funções, designs mais atrativos, mais poderosos e com mais funções, desde novos tipo de câmaras fotográficas, novas tecnologias de som, emissores de infravermelhos e novas tecnologias de comunicação, como o Bluetooth 4.0 ou o NFC.

Estas inovações apresentam cada vez mais capacidades e possibilidades para serem aplicadas em diversas situações do nosso dia a dia.

1.1 Contexto

A ideia deste estudo nasceu com base na ideia de se conseguir facilitar algumas tarefas na gestão de património através da distribuição da informação de cada peça de património em vez de, simplesmente, se limitar a atribuir um identificador a cada entidade e, as restantes informações dessa peça estarem guardadas numa base de dados.

Atualmente, nas organizações onde é aplicado um sistema de gestão de património estável, é atribuído a cada peça um código de barras que serve como identificador único dessa peça. Se alguém quiser saber mais informações dessa peça, como por exemplo, o departamento a que pertence ou qual o utilizador responsável, essa informação terá de ser consultada através da base de dados que contém essa informação. Contudo, essa informação poderá ser necessária num momento onde o acesso a essa base de dados não é possível.

Esse problema poderia ser resolvido através da utilização de etiquetas *NFC* que guardariam as informações necessárias da peça de património na própria chapa de património.

No entanto, a informação que fosse guardada, poderia ser considerada sensível e estaria acessível a qualquer pessoa com um leitor de *NFC*.

Desse modo e visto que, atualmente, a tecnologia *NFC* não permite qualquer tipo de cifragem, nasceu a necessidade de se fazer esta análise do comportamento destes algoritmos num dispositivo móvel, a fim da informação cifrada ser utilizada em etiquetas *NFC*.

O resultado deste estudo não fica limitado a este cenário, sendo possível utilizar esta ideia para fins de transmissão de mensagens de forma *offline* ou aplicado a jogos onde, por exemplo, uma determinada mensagem estaria escondida nas etiquetas e que só fosse possível ler tendo a chave de decifragem.

1.2 Objetivos do Trabalho

Neste trabalho pretendemos analisar os diferentes tamanhos e tempos que são obtidos comparando o comportamento de diferentes processos de compressão e cifragem em diversos formatos de ficheiros, num dispositivo móvel.

No final, será apresentada a melhor combinação de algoritmos de compressão e cifragem de modo a que seja possível implementar no futuro uma aplicação que garanta a escrita e proteção dos dados em etiquetas *NFC*.

O contributo que será dado com este trabalho será a análise do comportamento destes algoritmos, em diferentes situações, analisando o tamanho dos ficheiros resultantes e o tempo que é dispendido tanto no processo de cifragem, como no processo de decifragem da informação.

1.3 Estrutura do Documento

Este relatório, tem no total cinco capítulos. Sendo este capítulo 1 a introdução do trabalho e do relatório.

No capítulo 2 são descritos os diferentes conceitos e funcionamentos dos algoritmos utilizados neste trabalho, assim como trabalhos que foram considerados relevantes para o tema.

No capítulo 3 é apresentada a forma como os algoritmos escolhidos são analisados, assim como a informação que é utilizada para os testes e pontos considerados importantes na implementação desses mesmos testes.

No capítulo 4 são apresentados os resultados obtidos, assim como a análise desses resultados com base nas variáveis que são consideradas relevantes.

Finalmente, no capítulo 5 serão apresentadas as conclusões dos resultados obtidos com este trabalho, assim como a apresentação de sugestões a serem aplicadas em trabalho futuro.

Capítulo 2

Estado da Arte

Neste capítulo serão apresentados diversos conceitos que serão abordados neste trabalho.

Inicialmente será abordado a tecnologia NFC, de modo a se compreender as potencialidades e funcionamento desta tecnologia, contextualizando assim a motivação para este trabalho.

De seguida será abordado o tema da cifragem de informação, sendo abordado os diferentes tipos de algoritmos que serão usados no trabalho.

Finalmente será abordado a temática da compressão, explicando o seu conceito e com uma breve descrição dos algoritmos utilizados neste trabalho.

2.1 Near-Field Communication

O *Near-Field Communication* ou *NFC* é um conjunto de protocolos que já existe há mais tempo do que a maior parte das pessoas tem ideia. Este conjunto de protocolos permite que dois dispositivos comuniquem através de uma ligação sem fios, à semelhança das transmissões por radio-frequência, ou *RFID*, mas que, ao contrário deste, tem a particularidade de só funcionar a distâncias bastante pequenas.

O *NFC* foi aprovado como um *standard ISO* a 8 de dezembro de 2003. A 18 de março de 2004 a *Nokia Corporation*, a *Royal Philips Electronics* e a *Sony Corporation* criaram o Forum *NFC* como forma de promover o desenvolvimento, implementação e divulgação da tecnologia de forma a que fosse usada na interação entre dispositivos inteligentes e no desenvolvimento de novas formas de pagamento.

Este conjunto de *standards* permite vários tipos de comunicação entre pontos. Estes pontos podem ser considerados ativos ou passivos. Esses pontos são considerados ativos se estiverem a receber energia de uma fonte externa, como por exemplo um *smartphone*

ou um terminal *NFC*, enquanto que para ser considerado passivo, o ponto da comunicação não está a receber energia de nenhuma fonte. Um exemplo de um ponto passivo são as etiquetas, ou *tags NFC*, pequenos circuitos com uma antena.

O tamanho destas etiquetas é extremamente reduzido, não sendo comum encontrar etiquetas com capacidades superiores a 1 KB. Os tamanhos mais comuns e utilizados rondam entre os 64 Bytes e os 192 Bytes. Existem modelos de alta capacidade que chegam a ter entre 4 KB a 8 KB, mas revelam-se bastante caros, quando comparados a modelos com capacidade mais reduzida.

De forma a que seja estabelecida um canal de comunicação entre dois pontos, é necessário que pelo menos um deles seja um ponto ativo. Caso os dois pontos sejam ativos, é possível criar um canal onde ambos os pontos comunicam entre si, enviando e recebendo cada um, de forma independente, informação.

Numa comunicação ativo-passivo, a comunicação so é possível ser feita num sentido onde o ponto ativo lê ou escreve informação no ponto passivo. Esta é a forma como dispositivos *NFC* interagem com *tags NFC* onde é possível guardar pequenas quantidades de informação. Nesta situação o dispositivo transfere energia, através de indução magnética, para a *tag* conseguindo, desta forma, ler ou escrever informação no circuito.

Mesmo que a comunicação entre um dispositivo *NFC* e uma *tag* seja segura, a informação que fica guardada no circuito encontra-se disponível para qualquer outro leitor que consiga aproximar-se o suficiente. Não existe uma forma de, fisicamente, proteger a leitura da informação que lá fica guardada, por isso, esse problema deverá ser resolvido através de um método de mais alto nível. A única proteção fornecida por esta tecnologia é contra a escrita de dados, sendo este método irreversível.

2.2 Cifragem

cifragem é um processo criptográfico de codificação de informação, de tal maneira que qualquer pessoa que não saiba como o descodificar não o consegue compreender.

No processo de cifragem, a informação é convertida de um estado compreensível para outro incompreensível, onde ninguém o consegue compreender. A única maneira de se voltar a reaver a informação encriptada é sujeitando a mesma a um processo de decifragem onde a informação passa de novo a um estado compreensível.

Os algoritmos que definem como é que a informação é encriptada ou desencriptada são chamados de cifras. Geralmente estes algoritmos utilizam uma chave que é utilizada para codificar ou descodificar a informação podendo utilizar a mesma chave para fazer o processo inverso, como acontece nas cifras simétricas, ou necessitando de uma outra chave, como acontece com cifras assimétricas ou de chaves públicas e privadas.

2.2.1 Criptografia Simétrica

A criptografia simétrica, também conhecida como criptografia de chave secreta ou partilhada, é um tipo de criptografia que utiliza uma chave para encriptar e desencriptar a informação. Esse processamento da informação pode ser em fluxo ou em bloco.

Num processo em fluxo, a informação é processada conforme vai sendo enviada no canal de comunicação, sendo encriptada ou desencriptada durante o processo, geralmente bit a bit. Caso o processamento seja feito em bloco, a informação é dividida em pequenos segmentos com um determinado tamanho que são posteriormente processados.

Estes tipos de algoritmos são mais simples, extremamente mais rápidos e computacionalmente menos custosos quando comparados com algoritmos assimétricos.

O inconveniente deste tipo de algoritmo é que a chave que cifra/decifra a informação deverá ser conhecida por todos os intervenientes da comunicação, quer sejam emissores ou destinatários. Este facto faz com que a segurança do algoritmo dependa não só do algoritmo em si e da chave como também do modo como a chave é partilhada entre os intervenientes. Assim, na eventualidade da chave ser comprometida, qualquer pessoa poderá ter acesso a qualquer informação encriptada.

2.2.2 Criptografia Assimétrica

A criptografia assimétrica, também conhecida como criptografia de chave pública, é um tipo de criptografia que utiliza um conjunto de chaves necessárias para a cifragem ou decifragem. Existem duas utilizações principais para este tipo de algoritmo, cifragem de chave pública e assinatura digital.

A chave pública geralmente é distribuída livremente por todos aqueles com quem se pretende manter uma comunicação segura, enquanto que a chave privada é guardada pelo dono, não sendo transmitida a ninguém.

Numa situação em que se pretende trocar informação encriptada, a chave pública só pode ser usada unicamente para a cifragem da informação, não sendo possível desencriptar essa informação através da utilização da chave pública. A decifragem só pode ocorrer quando a chave privada está presente. Desta forma é garantido que todos podem encriptar a informação, mas só o destinatário, aquele que tem a chave privada, é que poderá desencriptar a informação. Desta forma, garante-se que é mantida a confidencialidade da informação, visto que nem o próprio emissor tem a capacidade de desencriptar a informação.

No caso da assinatura digital, a informação não é encriptada, mas é adicionado um bloco de texto que é gerado através da chave privada, com base na mensagem escrita. A autenticidade da assinatura poderá ser verificada com a chave pública, validando que foi escrita pelo dono da chave e, adicionalmente, garantindo que a mensagem não sofreu quaisquer alterações. Caso o texto assinado tenha sofrido qualquer alteração, quando o

destinatário for verificar a validade da assinatura através da utilização da chave pública do remetente, essa validação não irá estar correcta, indicando que o texto sofreu alterações.

Este tipo de criptografia é extremamente segura, especialmente porque a informação encriptada só fica acessível a quem tiver em sua posse a chave privada, não dependendo a sua segurança de mais ninguém a não ser do dono. Contudo, a sua utilização é mais complexa que outros tipos de criptografia e, na eventualidade da chave privada ter sido comprometida ou invalidada, obriga a um novo processo de criação de um novo conjunto de chaves e de uma nova distribuição da chave pública por todos os destinatários o que pode dar alguma confusão, especialmente quando esse número é bastante elevado. Adicionalmente, toda a informação encriptada com a chave antiga, fica invalidado, sendo impossível a sua decifragem. Computacionalmente é também mais pesado, quando comparado com algoritmos simétricos com segurança equivalente.

2.2.3 Cifras

Nesta subseção serão apresentados os conceitos dos algoritmos que serão utilizados e analisados neste trabalho.

A escolha destes algoritmos deve-se ao facto de serem bastante populares e alguns deles, apesar de já se encontrarem descontinuados e considerados pouco seguros, ainda são utilizados. Estes algoritmos encontram-se também já implementados e disponibilizados nas bibliotecas utilizadas para o desenvolvimento do trabalho.

As cifras simétricas apresentadas são *AES*(secção 2.2.3.1), *DES*(secção 2.2.3.2), *TripleDES*(secção 2.2.3.3) e *Blowfish*(secção 2.2.3.4). A cifra assimétrica utilizada é a *RSA*(secção 2.2.3.5).

2.2.3.1 Advanced Encryption Standard

O *Advanced Encryption Standard* ou *AES* foi anunciado pelo *NIST*, o Instituto Nacional de Standards e Tecnologia dos Estados Unidos da América, como um processo para escolher *um algoritmo de cifragem, não classificado, divulgado publicamente capaz de proteger informação governamental sensível até ao próximo século.*

Este processo foi anunciado em 1997 e o vencedor foi selecionado no ano 2000.

A cifra vencedora era originalmente chamada de *Rijndael* e foi desenvolvida por Joan Daemen e Vincent Rijmen. O algoritmo usado é um algoritmo de chave simétrica, o que significa que é possível encriptar e desencriptar informação utilizando a mesma chave.

A cifra *AES* pode ser vista como uma variante da cifra original devido ao facto de utilizar blocos com um tamanho fixo de 128 bits, enquanto que a original permitia blocos de qualquer tamanho compreendido entre 128 bits e 256 bits, múltiplos de 32.

Devido à sua robustez e segurança, o *AES* é uma das cifras de chave simétrica mais usadas em todo o mundo.

2.2.3.2 Data Encryption Standard

O algoritmo DES, *Data Encryption Standard* era o algoritmo predominante antes do actual AES.

Este algoritmo foi criado na década de 1970 pela IBM, tendo sido publicado em 1977 e oficializado como standard em 1979.

Este algoritmo era bastante seguro na época, apesar de terem sido detetadas algumas falhas teóricas que, apesar de existirem, mostravam-se bastante difíceis de serem exploradas. Atualmente, devido ao avanço tecnológico dos computadores, este algoritmo é considerado bastante inseguro devido ao facto de utilizar uma chave bastante pequena. No ano de 1998 a Electronic Frontier Foundation conseguiu quebrar a segurança da chave em 56 horas. Em 1999 o algoritmo foi publicamente demonstrado que era possível quebrar a segurança do algoritmo em menos de um dia. Em 2006, a Universidade de Bochum e Kiel, na Alemanha, conseguiu quebrar o algoritmo em 9 dias com um hardware de 10.000 dollars. Um ano depois, esse tempo foi reduzido para menos de 7 dias, sendo no ano seguinte reduzido para menos de um dia. [S. 06]

Acredita-se que o algoritmo seja praticamente seguro sobre a forma do TripleDES, abordado na secção 2.2.3.3, contudo, este não tem tido grande adesão, estando o algoritmo a ser ultrapassado pela cifra AES.

2.2.3.3 Triple Data Encryption Standard

O TripleDES é um algoritmo de cifra de bloco que foi criado como forma de se arranjar uma solução simples para o problema que o DES estava a enfrentar na época com o avanço no poder de processamento o que possibilitava ataques de força bruta mais rápidos.

Desta forma, evitando a necessidade de se desenhar um novo algoritmo, o TripleDES utiliza o algoritmo DES que executa três rotinas de cifragem com o algoritmo DES.

Cada uma destas rotinas pode ter a sua própria chave ou usarem todas a mesma chave, contudo, esta última opção não é recomendada do ponto de vista de segurança, pois o comportamento do algoritmo é igual ao DES.

2.2.3.4 Blowfish

A cifra *Blowfish* é uma cifra de bloco que foi criada em 1993 por Bruce Schneier e é atualmente uma das cifras mais usadas em todo o mundo, estando disponível num conjunto de produtos relacionados com cifragem.

Um dos principais motivos da popularidade deste algoritmo deve-se ao facto de ter sido criado como alternativa para a cifra *DES* e, ao contrário de outras cifras também

criadas na altura, foi colocado como domínio público de modo a que pudesse ser utilizado por qualquer pessoa em qualquer parte do mundo.

Até à data, não foi encontrada qualquer tipo de vulnerabilidade para esta cifra, contudo, é susceptível a ataques caso a chave utilizada seja fraca. Deste modo, os utilizadores desta cifra devem utilizar chaves mais fortes ou usar alternativas mais seguras. Bruce Schneier disse, em 2007, que achava surpreendente o facto de ainda utilizarem esta cifra, ainda mais por haver já sucessores como o *Twofish* e o *Threefish*[Dah07].

Estas duas cifras alternativas foram feitas pelo mesmo criador do *Blowfish*

2.2.3.5 RSA

O algoritmo RSA é um algoritmo assimétrico ou de chave pública que foi publicado em 1977 por Ron Rivest, Adi Shamir e Leonard Adleman.

Este algoritmo para funcionar necessita de uma chave pública para encriptar e uma chave privada para desencriptar a informação. Este algoritmo também pode ser utilizado para assinatura digital.

O funcionamento desta cifra é semelhante ao que está indicado na secção 2.2.2.

2.3 Compressão

Compressão de dados geralmente refere-se ao processo de comprimir e reduzir o tamanho de informação. Este processo baseia-se em analisar a informação que se pretende comprimir e aplica diversos processos que alteram essa informação, de acordo com o tipo de compressão que está a ser aplicada.

A compressão pode ser de dois tipos, com perda de dados ou sem perda de dados. No caso da compressão com perda de dados, a informação é analisada e os bits que forem considerados desnecessários são eliminados, reduzindo assim o tamanho do ficheiro. A informação eliminada não pode depois ser recuperada invertendo o processo. Este tipo de compressão é bastante usada para conteúdos multimédia, onde muita informação guardada nesses conteúdos não é geralmente vista pelo ser humano, como por exemplo, determinadas frequências em ficheiros de áudio.

No caso da compressão sem perda de dados, a informação é analisada e os bits que forem determinados como redundantes são abreviados de forma a que seja possível reverter essa informação. [Sol06]

O processo de compressão e descompressão da informação geralmente exige um esforço extra de processamento. Esse esforço poderá ser efetuado num processo de descompressão *offline*, onde a informação é descomprimida e reposta ao seu estado inicial antes de ser lida, ou é descomprimida de modo *online*, onde a descompressão está a ser executada ao mesmo tempo da leitura da informação.

Os algoritmos que serão utilizados neste trabalho são o algoritmo de Lempel-Ziv-377 Markov, DEFLATE e BZip2, descritos de seguida.

LZMA

O algoritmo de Lempel-Ziv-Markov, ou LZMA, é um algoritmo de compressão, sem perda de dados, que se encontra em desenvolvimento desde 1998[Sol06].

Este algoritmo utiliza um esquema de compressão baseado em dicionário, ou seja, procura padrões na informação, organizando esses padrões numa lista e substituindo o padrão no ficheiro original, por uma referência para a posição do padrão nessa lista.

DEFLATE

O algoritmo DEFLATE é um algoritmo de compressão que utiliza uma combinação de vários métodos de compressão.[Sol06]

Atualmente este algoritmo está definido de forma a não estar sujeito a qualquer tipo de patentes, tornando-o bastante popular para diferentes implementações de outros formatos, como por exemplo, o formato de imagem PNG.

BZip2

O algoritmo BZip2 é um algoritmo de compressão de informação, *open-source* que foi publicado inicialmente em 1996.[Sol06]

Ao contrário de muitos algoritmos de compressão, este algoritmo só permite a compressão de um único ficheiro, não sendo possível comprimir múltiplos ficheiros.

Comparativamente aos outros algoritmos de compressão, o BZip2 é considerado mais eficiente que o DEFLATE, contudo é também mais lento.

Capítulo 3

Avaliação dos Algoritmos

Nos dias de hoje, cada vez mais dispositivos móveis estão a ser usados como verdadeiros assistentes para a nossa vida pessoal e profissional. Com a evolução tecnológica é cada vez maior a quantidade de informação que é guardada nestes dispositivos, são criados novos modelos com mais funções, designs mais atrativos, mais poderosos e com mais funções, desde novos tipo de câmaras fotográficas, novas tecnologias de som, emissores de infravermelhos e novas tecnologias de comunicação, como o Bluetooth 4.0 ou o NFC.

Nesta época de rápida evolução tecnológica, a maior parte das pessoas não conseguem acompanhar nem ganhar conhecimento sobre determinadas práticas que deverão ter sobre a utilização destas tecnologias. Ainda não há muito tempo o número de computadores ligados à Internet era tão reduzido que não havia grande necessidade para elaborados protocolos de segurança. Quando os primeiros computadores pessoais começaram a aparecer no mercado, a sua utilização era tão simples que a maior parte das pessoas não guardava qualquer informação sensível neles, desvalorizando as práticas de segurança. Contudo, nestes últimos anos as pessoas foram cada vez mais depositando informação bastante sensível sem se lembrarem que, neste mundo cada vez mais ligado, é necessário aumentar as práticas de segurança.

Com esta evolução e aparecimento de novas tecnologias, uma que se destaca pela sua simplicidade e potencial é a Near Field Comunication, também conhecido como NFC. Este conjunto de normas permite a transferência de informação, entre dois pontos, desde que estes se encontrem a uma distância bastante curta. Esta tecnologia é semelhante a outras tecnologias de radio-frequência, contudo a maneira como está construída faz com que seja extremamente difícil de interceptar a comunicação entre dois pontos sem que seja evidente para pelo menos um dos intervenientes da comunicação.

Devido a estas propriedades o NFC está a ser visto como uma forma segura de se efetuar comunicações sensíveis, como para se efetuar pagamentos através da utilização

de smartphones. O facto de só ser possível dois dispositivos comunicarem estando bastante próximos um do outro permite também que esta tecnologia seja aplicada como um novo método para emparelhar dois dispositivos para uma ligação direta evitando todo o processo e a necessidade de se efetuar uma pesquisa de dispositivos e troca de códigos. É garantido que para dois dispositivos comunicarem entre si por NFC tiveram de estar em contacto.

Outra particularidade desta tecnologia é o facto de ser uma tecnologia de baixo consumo energético o que possibilita a transferência de energia por indução magnética. Isto permite que exista uma transmissão de informação entre dois pontos, mesmo que um desses pontos não tenha qualquer fonte de energia. Este é um caso comum com a utilização de etiquetas ou *tags* NFC. Estas etiquetas tem um pequeno circuito onde é possível guardar informação, assim como uma antena que consegue receber energia através de indução magnética, o que permite que esta receba energia de um dispositivo NFC e que a sua informação seja lida, sem a necessidade de estar ligada a qualquer fonte de energia.

A utilização destas etiquetas permite um largo conjunto de novas aplicações. Entre elas, a sua utilização para identificação, onde cada etiqueta seria a identificação para cada entidade, utilização para *smart posters* com aplicação semelhante à atualmente dada aos códigos QR, mas sem a dependência do espaço físico nem da qualidade da câmara do dispositivo e ainda a possibilidade de se automatizar algumas tarefas no smartphone utilizando as etiquetas como marcadores de localizações, por exemplo, colocar o telefone em silêncio quando entra em contacto com uma etiqueta localizada no local de trabalho.

Visto não haver qualquer tipo de limitação quanto ao tipo de informação que pode ser guardada nestas etiquetas, as aplicações podem ser para qualquer finalidade. Inclusive, podem ser utilizadas para guardar mensagens de forma *offline* ou até serem utilizados para guardarem informação sobre um determinado património de uma empresa, algo semelhante ao já utilizado nos dias de hoje com códigos de barras. Contudo, a informação possível de armazenar num código de barras unidimensional é bastante limitada, sendo geralmente utilizado para guardar um número de identificação que depois é consultado numa base de dados onde estão as informações sobre o bem. Mesmo utilizando um código de barras bidimensional, para se guardar 512 Bytes de informação, iria ser necessário um código quadrado com 462 pixeis de lado.

Contudo, na eventualidade de não se ter acesso naquele momento à base de dados, não existe qualquer possibilidade de se obter informações sobre o bem em questão. Isso poderia ser resolvido através da utilização de códigos de barras bidimensionais, como os códigos QR. Contudo, quanto mais informação fosse guardada no código QR maior seria o tamanho do padrão e, consequentemente a qualidade do leitor do código teria de ser bastante superior, não podendo ser utilizado qualquer dispositivo com uma câmara. Como referido no paragrafo anterior, um código QR que armazenasse cerca de 512 Bytes de informação iria ter 462 pixeis por 462 pixeis, obrigando a que a etiqueta com este

código tivesse dimensões bastante grandes ou que o dispositivo de leitura tivesse uma elevada precisão.

Caso se utilizassem etiquetas NFC, o tamanho físico da etiqueta será sempre independente da informação guardada, pode ser lida por qualquer dispositivo com capacidade de leitura NFC e, ao contrário dos códigos de barras, pode ter a informação reescrita sem a necessidade de substituição da etiqueta.

Esta facilidade possibilita que para além de ser guardado o número de património do bem, possam ser guardadas outras informações como o departamento a que estaria associado, o utilizador responsável, ano de aquisição, entre outras informações que fossem relevantes para a empresa. Mas apesar de ser possível guardar estas informações nas etiquetas e da comunicação entre o leitor e a etiqueta não ser fácil de interceptar, é possível a qualquer pessoa com um leitor ler a etiqueta posteriormente e ter acesso a toda a informação que esteja guardada nela. Não existe atualmente qualquer tipo de sistema de cifragem integrado nas atuais etiquetas NFC. Assim, essa tarefa terá de ser executada pela aplicação que estará a guardar a informação.

O objetivo principal da cifragem é tornar a informação ilegível para qualquer entidade que não tenha acesso à chave necessária para a conseguir ler. Mas esse processo troca a performance e espaço pela segurança. Atualmente as etiquetas NFC têm um espaço bastante limitado para se guardar informação, não sendo comum chegar a 1 KB, e o processo de cifragem não deverá demorar muito tempo a ser efetuado de modo a ser possível ser processado em quase tempo real num dispositivo móvel.

Neste trabalho serão analisados diversos tipos de cifras e serão comparados os seus desempenhos tanto em termos de tempo como em termos de tamanho necessário para o seu processamento em dispositivos móveis.

3.1 Equipamento Utilizado

Neste trabalho foi desenvolvida uma simples aplicação que corre num smartphone Android, executando o processo de cifragem e decifragem de vários tipos de dados, indicando no final o tempo dispendido para cada operação, assim como o tamanho final da informação antes e depois da decifragem.

Foi escolhida a plataforma Android, devido ao facto de ser uma das plataformas mais populares no mercado, assim como é das plataformas mais abertas e compatíveis, não exigindo qualquer tipo de equipamento ou software específico.

Outro fator de decisão para a escolha da plataforma, deve-se ao facto de, atualmente, o maior número de dispositivos com capacidade NFC são dispositivos Android. Ainda mais, as cifras escolhidas para os testes nas aplicações encontram-se já implementadas nas bibliotecas do SDK utilizado, facilitando a sua utilização e reduzindo assim eventuais erros que possam, de certa forma, adulterar os resultados obtidos.

O dispositivo utilizado, um Samsung Nexus S, tem como sistema operativo o Android 4.1.2, tendo um processador de um único núcleo de 1 GHz e com 512 MB de memória RAM.

A aplicação foi desenvolvida com a versão 15 do SDK do Android, o que corresponde à versão 4.0.3 Icecream Sandwich.

A aplicação foi desenvolvida para trabalhar sobre dados binários, de forma a que se poderia testar a aplicação sobre diferentes tipos de informação.

Desta forma, foram utilizados seis tipo de ficheiros, com diferentes características e extensões, de forma a que fossem passiveis de serem guardados em etiquetas NFC.

De forma a facilitar a sua referência, os ficheiros utilizados serão identificados por um identificador. os ficheiros utilizados estão identificados na tabela 3.1.

Referência	Descrição
BMP	Imagem em formato bitmap, com uma matriz de 6px por 6px, monocromática.
JSON	Ficheiro com informação de texto genérica em formato JavaScript Object Notation.
PNG	Imagem em formato Portable Network Graphics, com uma matriz de 6px por 6px, monocromática.
TXT	Pequeno ficheiro de texto com alguns caracteres acentuados.
TXT_512B	Ficheiro de texto grande com 512 Bytes de texto aleatório.
XML	Ficheiro com informação genérica no formato eXtensible Markup Language.

Tabela 3.1: Ficheiros utilizados para os testes

Os ficheiros escolhidos para os testes foram os considerados mais adequados para serem guardados nas actuais etiquetas NFC, devido ao seu relativo pequeno tamanho e à versatilidade da informação que podem conter.

Estes ficheiros podem ser também organizados em três grupos, como identificados na tabela 3.2

Grupo	Ficheiros
Imagens	BMP
	PNG
Notação	JSON
	XML
Texto	TXT
	TXT_512B

Tabela 3.2: Grupos de ficheiros

O grupo Imagens, inclui dois formatos de imagem, bastante populares que armazenam bastante informação sobre a imagem, evitando qualquer tipo de perda de informação, ideal para se armazenar informação sensível, como códigos de barras unidimensionais e bidimensionais.

O segundo grupo, Notação, diz respeito a ficheiros de texto num formato de notação, neste caso JSON e XML. Este tipo de notações são feitos com o proposito de, para além de guardar informação, tornar a leitura dessa informação mais fácil para o ser humano, sendo fácil compreender o seu conteúdo sem a necessiade de um software especial. Cada ficheiro contém informação genérica, possível de ser armazenada em etiquetas NFC.

O último grupo, Texto, diz respeito a ficheiros elementares de texto que poderão guardar qualquer tipo de informação em formato texto, não estando limitado a qualquer tipo de regras, ao contrário do que acontece com os ficheiros de Notação. Esta forma permite uma maior liberdade para se organizar a informação da forma mais conveniente, contudo, podendo dificultar a sua leitura, aumentar desnecessáriamente a sua complexidade ou impossibilitando o acesso da informação a qualquer outra aplicação que não conheça a estrutura do ficheiro. São usados dois ficheiros semelhantes, com tamanhos diferentes. Um dos ficheiros contém muito pouca informação, enquanto que o segundo já contém 512 Bytes de informação que é mais que a maior parte das etiquetas NFC actuais conseguem suportar.

3.2 Implementação

Para se efetuarem os testes foi feita uma aplicação Android que efetua as operações de cifragem e decifragem num dispositivo Android, utilizando diferentes cifras, medindo o tempo gasto entre cada uma dessas operações, assim como o tamanho do ficheiro obtido.

É sugerida, na imagem 3.1, a arquitetura de funcionamento da aplicação.

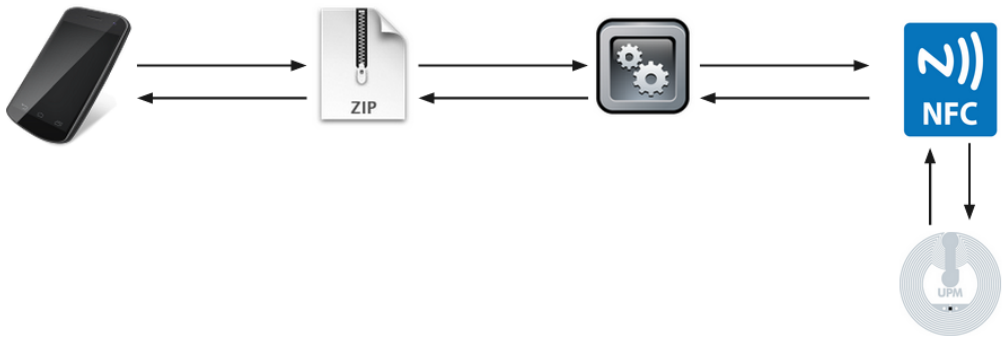

Figura 3.1: Arquitetura de Funcionamento

A informação é inserida no dispositivo móvel, através da inserção da informação pela interface gráfica ou por ficheiro. Por sua vez, essa informação é comprimida, cifrada e transferida para a interface NFC que, por sua vez, guardará essa informação na etiqueta NFC pretendida.

Quando se pretender proceder à leitura da informação contida nessa etiqueta, é invertido o processo, sendo a informação lida pela interface NFC e, posteriormente decifrada e descomprimida, sendo apresentada a informação lida na interface do dispositivo.

A aplicação foi desenvolvida como sendo uma plataforma de testes em dispositivos móveis, sendo depois adaptada de forma a se obter uma interface mais simples com a finalidade de poder ser utilizada como uma aplicação final.

Tentou-se que a construção da aplicação fosse modular de forma a facilitar a construção e posterior modificação da aplicação, estando também de acordo com a filosofia de construção de aplicações Android.

Foi idealizado uma linha de processamento onde o ficheiro percorreria um conjunto de fases de modo a ser obtido o ficheiro final encriptado. O ficheiro é inicialmente comprimido, sendo gerado o ficheiro comprimido, por sua vez, esse ficheiro é cifrado com base numa chave com as definições da cifra, resultando no final um ficheiro encriptado. Um simples esquema do processo pode ser visto na figura 3.2. De forma análoga, o processo inverso de decifragem ocorre invertendo esta sequência.

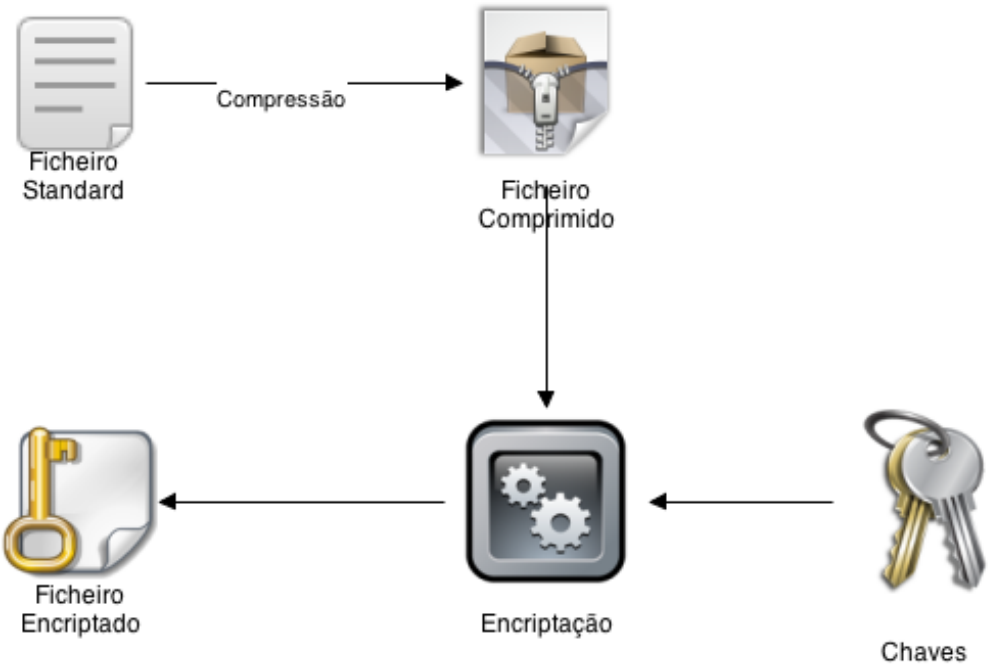

Figura 3.2: Processo de cifragem

Para o funcionamento da aplicação, foram criadas duas classes principais de forma a funcionarem como os blocos principais da aplicação. A primeira classe, *NFSecure Bundle* é usada como abstração para o tipo de informação que poderá ser usada. É onde é armazenada a informação a ser processada posteriormente. Esta classe está feita de modo a que consiga guardar a informação antes e depois de ser encriptada, sendo sempre

possível de aceder a qualquer um dos dois tipos de informação, sem a necessidade de se repetir de novo o processo de cifragem.

A classe, *NFSecure Crypto Engine* é a responsável pelas operações de cifragem e decifragem. Quando a classe é criada é indicada a cifra que se pretende que ela assuma, sendo inicializado todos os componentes necessários de acordo com as especificações da cifra. Esta classe tem depois a capacidade de receber um *Bundle*, aplicando a operação que se pretende e guardando a informação resultante.

Estas duas classes foram depois usadas para a criação das classes de teste. As classes resultantes, *NFSecure Testing Bundle* e *NFSecure Testing Crypto Engine*, funcionam de forma semelhante, contudo organizam-se em forma de matriz. No caso do *Testing Bundle*, a informação guardada corresponde não só à informação original, como aos dados obtidos após a cifragem e os dados obtidos após a decifragem, permitindo verificar se o resultado obtido após a decifragem correspondia à informação original antes da cifragem.

Adicionalmente, foi adicionada uma nova matriz onde seriam guardadas as informações relativas ao tempo de execução, tanto do processo de cifragem como de decifragem. Essa informação era obtida, analisando o tempo que foi dispendido desde a chamada do processo de cifragem ou decifragem, até ao final dessa chamada. Durante esta recolha de tempo, o processo de cifragem era repetido umas milhares de vezes, de forma a reduzir o erro de medição do tempo. No final o tempo era guardado em milisegundos, sendo posteriormente recolhido e analisado.

Relativamente à classe *Testing Crypto Engine*, foram guardadas as diferentes classes iniciadas para cada uma das cifras a serem utilizadas para os testes.

O processo da rotina de testes pode ser facilmente visualizado através da figura 3.3.

De forma a aplicar também boas regras de criptografia, a chave usada nas cifras simétricas é processada antes de ser utilizada no processo de cifragem, não sendo usado somente uma string como chave.

A este processo chama-se *salting* e tem como objetivo criar uma maior entropia na chave. O processo baseia-se em ter um conjunto de dados aleatório que é posteriormente adicionado à password inserida pelo utilizador. Com base nessa combinação é gerada uma hash que servirá para a criação da chave de cifragem.

O bloco de código utilizado para a operação de *salting* é o seguinte.

```
//SALT - Valor constante para acrescentar entropia na password
String SALT = "t5cCQM9drGx2Bbj3Cu6WMR3gp9cPJ5mdvA";

//Combina o SALT com a password do utilizador
byte[] combined_password = (SALT+password).getBytes();
```

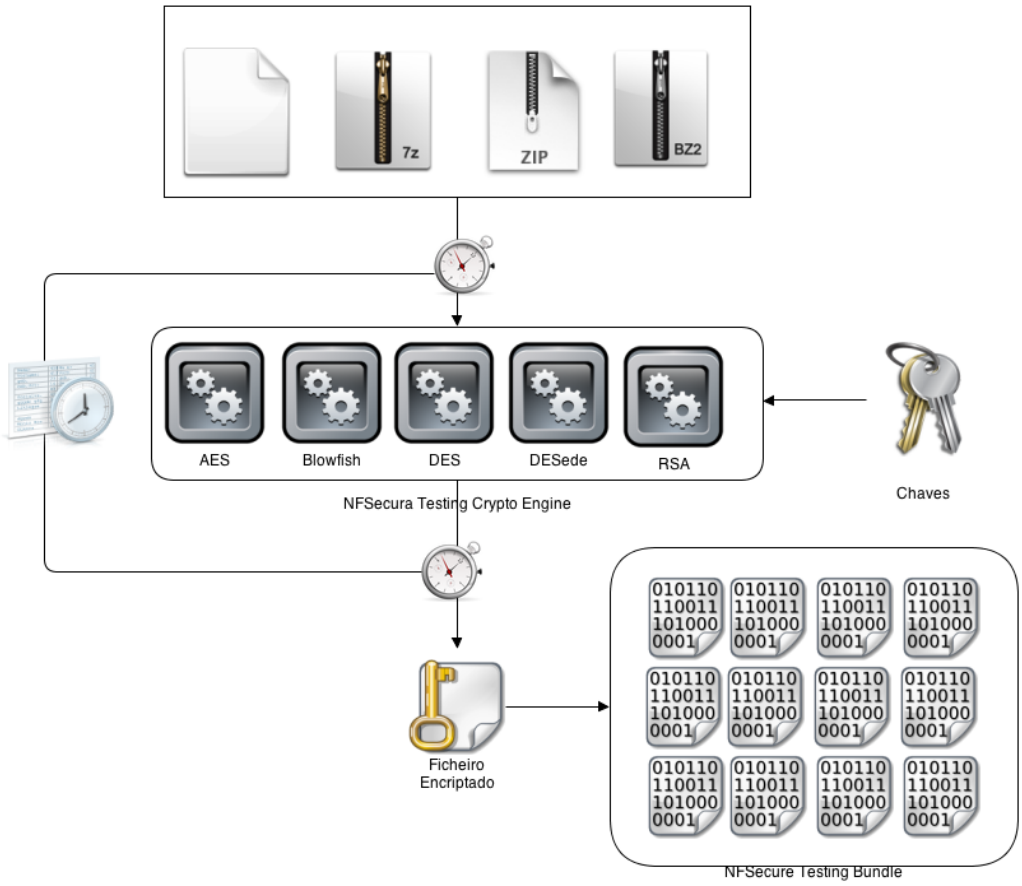

Figura 3.3: Rotina de Testes - cifragem

```
//Inicia o tipo de Hash pretendida
MessageDigest sha = MessageDigest.getInstance("SHA-256");

//Gera a hash da password utilizada
byte[] cryptoKey = sha.digest(combined_password);
```

Neste caso, obtemos no final uma hash de 256 bits, podendo alterar posteriormente o tamanho da chave usado internamente à aplicação e mantendo a segurança das chaves independente do tamanho da password introduzida pelo utilizador.

Este processo garante também que, sempre que a password que o utilizador inserir for a mesma, o resultado obtido pelas operações de *salting* e hashing são sempre os mesmos.

Este sistema é, no entanto, apenas usado pelas cifras simétricas, sendo a cifra RSA abordada de uma forma diferente. Neste caso, quando se tratava de criar as chaves para serem utilizadas com a cifragem assimétrica, limitou-se a criar um par de chaves pública e privada aleatórias, sendo estas posteriormente guardadas na memória do dispositivo onde estivesse a ser executado. Decidiu-se para este projecto utilizar chaves com um tamanho de 4096 Bytes para a encriptação assimétrica. Foi decidida a utilização deste tamanho de forma a que o tamanho do ficheiro cifrado fosse igual ao maior ficheiro utilizado nos testes, ou seja 512 Bytes, evitando o alinhamento dos blocos de cifragem.

Quando é inciado um processo de cifragem com a cifra RSA, a aplicações vai à localização definida e verifica a existência da chave pública na memória do dispositivo. Caso a chave exista, é usado essa para as operações, caso contrário, é gerado tanto uma chave pública como uma chave privada que são armazenas e utilizadas posteriormente nas operações.

No processo de decifragem, acontece algo semelhante, sendo necessário que a chave privada esteja no dispositivo para que o processo possa ser concluído. Caso a chave não exista no processo de decifragem, a aplicação não consegue avançar no processo, visto não haver maneira de se desencriptar a informação de chaves diferentes.

Foi também criada uma classe responsável pela compressão e descompressão da informação. Esta classe ao ser criada, é-lhe indicado o algoritmo de compressão que deverá utilizar, sendo iniciada com o que for necessário para se proceder às operações. Tem depois duas únicas funções públicas que tratam da compressão e descompressão da informação, recebendo a informação num modo, devolvendo noutro modo. Aplicando este tipo de abstração, possibilita que novos algoritmos sejam implementados e utilizados com a aplicação.

Com base na aplicação desenvolvida para os testes de performance, foi desenvolvida um pequeno protótipo onde o utilizador insere a informação que pretende processar e a

chave que pretende utilizar. Um protótipo da interface de introdução de dados pode ser vista na figura 3.4.

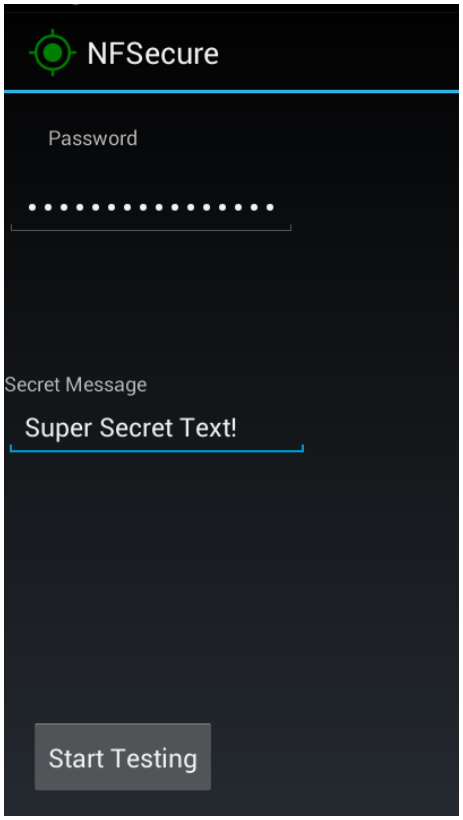

Figura 3.4: Protótipo de Interface - Introdução de Dados

Após isso, a informação inserida pelo utilizador é comprimida, e encriptada por todas as cifras simétricas disponíveis no dispositivo. A informação dessa cifragem é guardada e posteriormente é efetuada a decifragem e descompressão da informação. No final são apresentadas no dispositivo as informações relativamente ao tamanho dos ficheiros após cada processo de cifragem, assim como os tempos gastos para cada processo de compressão e cifragem. Um protótipo da interface com essa informação pode ser vista na figura 3.5.

3.3 Conclusões

A implementação desta plataforma de testes foi feita de forma bastante modular, permitindo que facilmente se consiga alterar a linha de processamento, ou a forma como determinadas classes funcionam.

Foram efetuados diversos testes durante a implementação de forma a garantir que os processos de armazenamento da informação, cifragem e recolha de variáveis estavam a

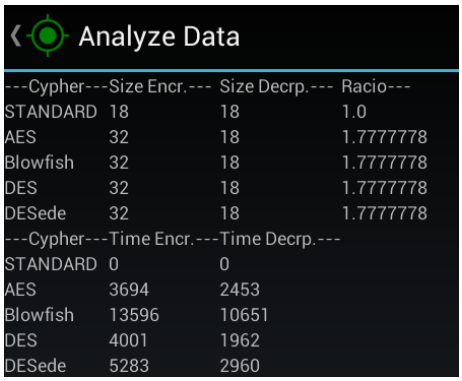

Figura 3.5: Protótipo de Interface - Resultado dos Dados

funcionar corretamente. No final é possível com esta aplicação de testes recolher informação sobre os tamanhos dos ficheiros resultantes e os tempos de execução para os processos de cifragem e decifragem da informação.

Capítulo 4

Análise de Resultados

Nesta seção são apresentados os testes efetuados na aplicação descrita e os dados que são recolhidos com base na rotina de testes criada.

Dado que os algoritmos usados são determinísticos, para cada conjunto de algoritmo e dados usados, o resultado produzido é sempre o mesmo. Desta forma, bastaria executar uma única vez o processo de forma a verificar os dados obtidos. No entanto, para se conseguir analisar os tempos de execução de cada processo, cada combinação foi testada, repetindo o processo um elevado número de vezes (5000 para cifras simétricas e 50 para assimétrica), sendo posteriormente calculado um tempo médio de execução, reduzindo eventuais interferências de outros processos que possam estar a correr no dispositivo de testes.

Apesar das diferenças serem extremamente pequenas, foram corridos três vezes as rotinas de testes, sendo agora apresentada a média dos resultados obtidos nessas repetições.

Em termos das unidades apresentadas, o tamanho dos ficheiros está expressa em Bytes enquanto que o tempo encontra-se expresso em milisegundos.

4.1 Testes Efetuados

Os testes efetuados basearam-se na execução da rotina de testes criada.

Esta rotina acede ao ficheiro a ser processado e converte-o para informação binária. Essa informação é posteriormente armazenada numa matriz de testes onde depois é encriptada e posteriormente desencriptada.

O processo é repetido para todos os ficheiros e toda a informação sobre o seu tamanho e tempo de execução é guardada e apresentada no final da rotina.

Para além de ser executada a rotina de cifragem/decifragem sobre os ficheiros, é também executada a compressão desses ficheiros e posterior cifragem/decifragem de forma a poder ser analisado os efeitos dessa compressão sobre o resultado final.

Para cada um dos seis ficheiros utilizados, é aplicada três diferentes tipos de compressão, baseados nos algoritmos de compressão LZMA, DEFLATE e BZip2, resultando em quatro modos de compressão para cada ficheiro, Standard, LZMA, DEFLATE e BZip2, totalizando assim vinte e quatro rotinas com os respetivos tamanhos e tempos de cifragem e decifragem.

A chave utilizada nos testes é sempre a mesma, evitando que influencie o resultado dos testes. A única exceção é na utilização da cifra RSA devido ao facto de ser um algoritmo assimétrico o que obriga à utilização de um conjunto de chaves pública e privada. Para tal, é gerado um par de chaves que são guardadas na memória do dispositivo e usadas para a cifragem e decifragem dos ficheiros. Para estes testes foi também utilizado o mesmo par de chaves para todos os ficheiros.

De forma a reduzir ao máximo o erro na medição do tempo de execução dos processos de cifragem e decifragem da informação, para cada ficheiro foi executado cada um dos processos 5000 vezes. Este método foi utilizado para todas as cifras, excluindo a cifra de RSA que foram feitas menos repetições. Isto deve-se ao facto desta cifra estar a demorar cerca de 2 a 3 segundos a ser executada, por processo, tendo havido a necessidade de reduzir o número de repetições para 50.

Os resultados obtidos são apresentados e analisados no capitulo seguinte.

4.2 Resultados Obtidos

O processo de testes implica a compressão da informação, seguida da sua cifragem e armazenamento na matriz de testes. Após isso é executada uma decifragem seguida de uma descompressão da informação. No entanto, os tempos registados dizem unicamente respeito aos processos de cifragem e decifragem, não estando contemplados os tempos gastos na compressão e descompressão da informação. A compressão foi feita antes da rotina de testes visto não se encontrarem atualmente implementados na aplicação. Recorreu-se a um programa de compressão de forma a se comprimir os ficheiros, resultando em ficheiros separados.

4.2.1 Tamanhos dos ficheiros

Na tabela 4.1 podemos ver o tamanho dos ficheiros quando não são sujeitos a qualquer tipo de compressão ou cifragem.

	Tamanho
BMP	86
JSON	432
PNG	138
TXT	110
TXT 512B	512
XML	198
Média	246

Tabela 4.1: Tamanho dos Ficheiros Usados (Bytes)

Na figura 4.1 estão apresentados os tamanhos médios dos ficheiros antes e depois de terem sido encriptados através de cada uma das cifras e com os diferentes tipos de compressão.

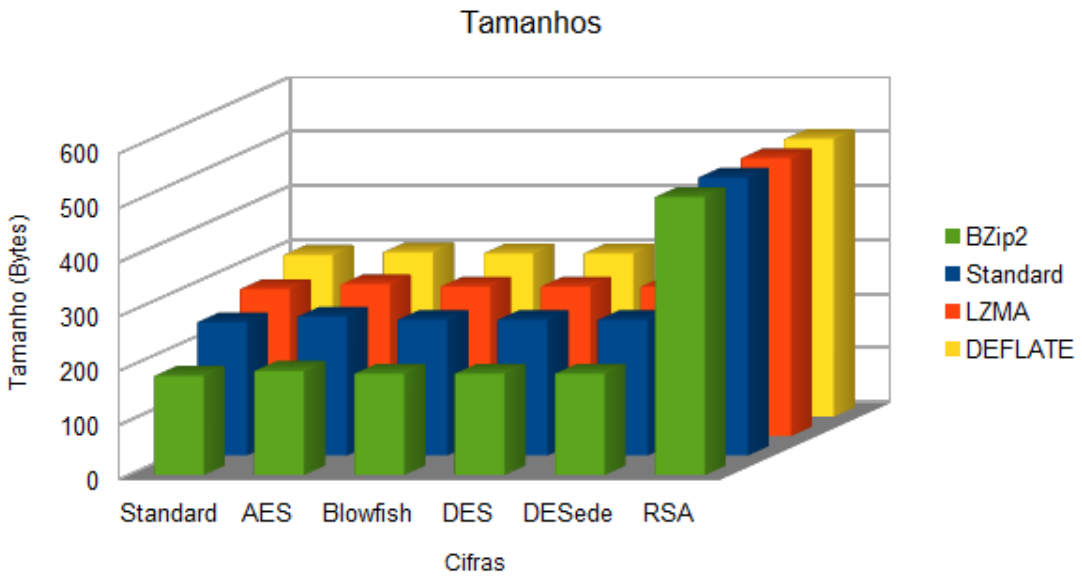

Figura 4.1: Tamanhos Médios Obtidos (Tabela A.1)

É possível verificar os quatro tipos de compressão utilizados, Standard para a compressão nativa do tipo do ficheiro, ou seja, sem qualquer compressão aplicada, LZMA, DEFLATE e BZip2.

É imediato verificar que a cifra RSA gera sempre ficheiros com o mesmo tamanho de 512 Bytes, não havendo qualquer tipo de alteração no que diz respeito ao tipo de compressão aplicado.

Algo que também é verificado é que, após ter sido aplicada a cifragem, o ficheiro resultante é sempre superior ao original, com ou sem compressão. Tal deve-se ao facto de se estar a utilizar cifras por blocos onde não foi feito um processamento dos blocos

cifrados.

4.2.2 Tempos de cifragem e decifragem

Foi também guardada a informação relativamente ao tempo de execução da função de cifragem e decifragem. Os dados obtidos para o processo de cifragem são apresentados no gráfico da figura 4.2 enquanto que os dados do processo de decifragem são apresentados no gráfico da figura 4.3.

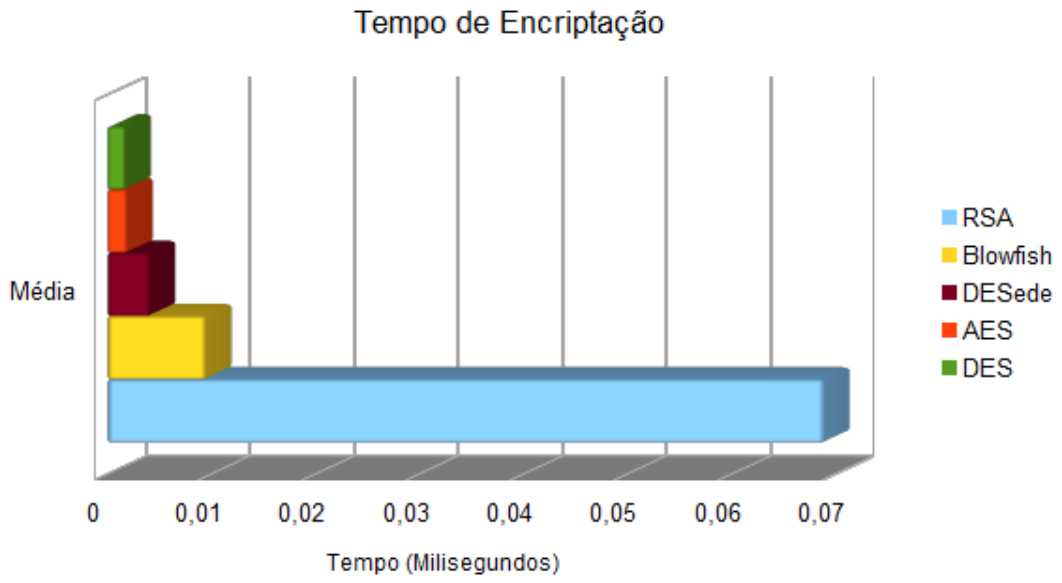

Figura 4.2: Tempos de cifragem por Byte (Milisegundos) (Tabela A.2)

Analisando os dados obtidos no processo de cifragem, é fácil de verificar que a cifra RSA será a que mais tempo demora a ser executada, quando comparada com as restantes. O tempo de execução desta cifra é quase sete vezes superior à cifra Blowfish que mostrou-se como a segunda com piores resultados. O resultado obtido pela cifra Blowfish deve-se não ao facto da cifra ser efectivamente mais lenta no seu processamento, mas sim à forma de como ela processa a inicialização das chaves internas ao processo.

A cifra DESede apresentou valores superiores à cifra DES, conforme esperado, visto serem a mesma operação, onde numa das cifras essa operação é repetida três vezes.

As cifras AES e DES apresentaram tempos de cifragem muito semelhantes, tendo a cifra DES ficado ligeiramente à frente.

Relativamente aos dados obtidos na fase da decifragem, numa primeira análise, a cifra RSA ficou bastante à frente em termos de tempo necessário, chegando aos quatro milisegundos por byte encriptado, enquanto que as restantes parecem todas terem o mesmo valor.

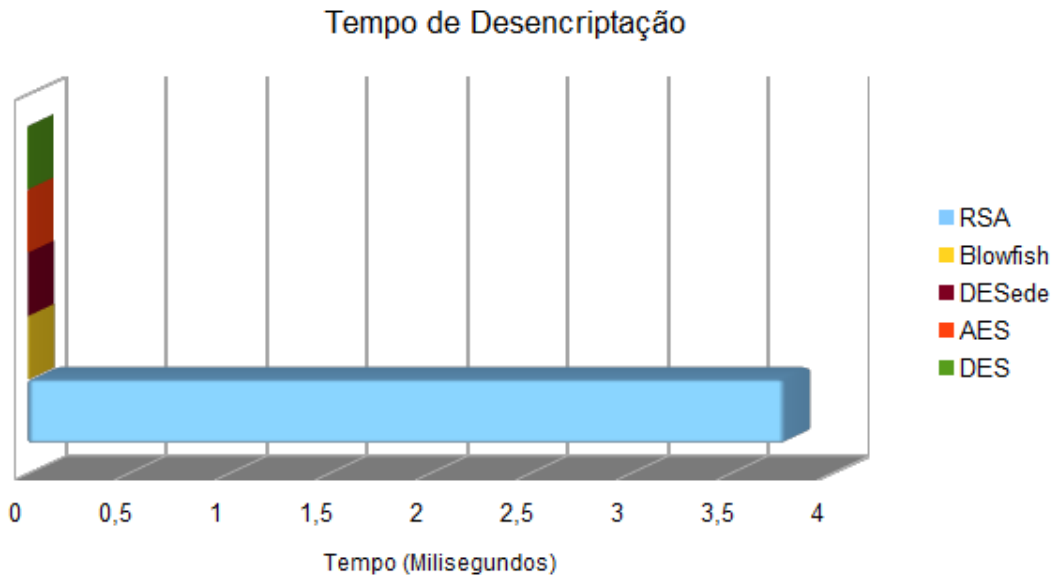

Figura 4.3: Tempos de decifragem por Byte (Milisegundos) (Tabela A.3)

Removendo a informação relativa à cifra RSA, podemos ver na figura 4.4, dados mais detalhados quanto à performance das cifras.

À semelhança do que aconteceu com o processo de cifragem, a cifra Blowfish foi a segunda mais lenta, seguida da DESede, AES e finalmente DES.

4.3 Análise

Analisando os valores médios dos tamanhos dos ficheiros após terem sido aplicada as diferentes compressões, temos os seguintes dados apresentados na tabela 4.2.

Compressão	Tamanho (Bytes)	(%)
Standard	294,33	100,00%
LZMA	315,25	107,11%
DEFLATE	336,39	114,29%
BZip2	241,14	81,93%

Tabela 4.2: Análise de Compressões

É possível ver que na maior parte dos casos, o resultado obtido após ter sido aplicada a compressão, são entre 7% a 14% maiores que o ficheiro sem compressão.

A única compressão que se mostrou vantajosa nos ficheiros utilizados para os testes foi a BZip2. Na tabela 4.3 é possível ver que só em dois casos é que o resultado obtido após a compressão é substancialmente superior ao ficheiro sem compressão.

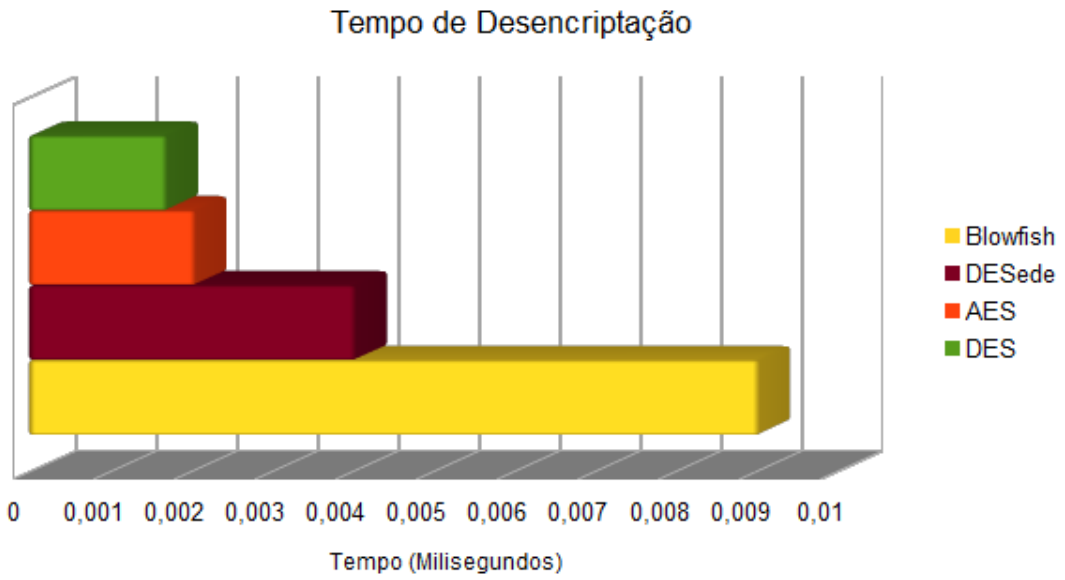

Figura 4.4: Tempos de decifragem por Byte sem RSA(Milisegundos) (Tabela A.3)

Ficheiro	Standard	BZip2	(%)
BMP	86	71	82,56%
JSON	432	226	52,31%
PNG	138	186	134,78%
TXT	110	135	122,73%
TXT 512B	512	316	61,72%
XML	198	163	82,32%

Tabela 4.3: Performance da Compressão BZip2

Em todos os outros casos, os resultados são consideráveis, permitindo que a informação possa ser guardada em etiquetas de capacidades mais baixas.

Vendo esta informação no gráfico 4.5, é possível ver que entre os 100 Bytes e os 150 Bytes a compressão não é vantajosa, sendo o ficheiro resultante após a compressão maior que o ficheiro sem compressão. No entanto, a partir desse valor e dentro dos tamanhos analisados, a compressão obteve sempre resultados vantajosos, chegando a valores de compressão de quase 50%.

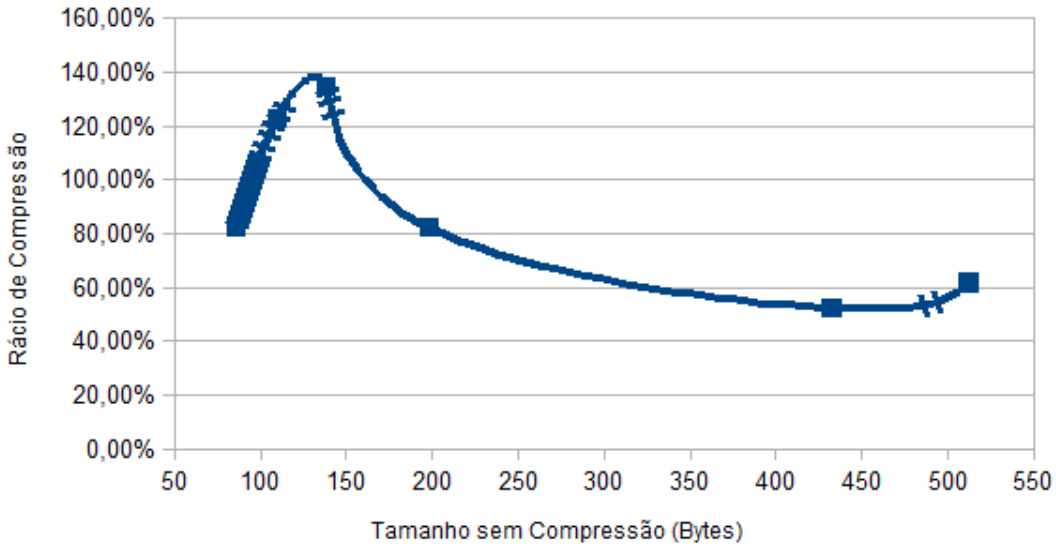

Figura 4.5: Compressão BZip2 (Tabela 4.3)

Assim, para este método de compressão pode ser ponderado a sua utilização para maiores quantidades de dados, permitindo que a informação seja guardada em etiquetas que tenham menos capacidade que a informação que se pretende guardar.

Analisando os diferentes algoritmos de compressão com base no tipo de ficheiro comprimido, podemos verificar os resultados no gráfico da figura 4.6.

É fácil desta forma verificar que o algoritmo de compressão BZip2 tem o maior número de casos vantajosos, enquanto que as restantes cifras não apresentam tantas vantagens.

Vendo pelo tipo de ficheiro, os únicos dois ficheiros que mostram serem bem comprimidos por todos os algoritmos são o ficheiro JSON de 432 Bytes e o TXT de 512 Bytes, mostrando que quanta mais informação o ficheiro tiver, maior a probabilidade de se conseguir efectuar uma compressão eficiente.

Os dois ficheiros com menos taxas de sucesso na compressão são o PNG de 138 Bytes e o TXT de 110 Bytes. O ficheiro mais pequeno, o BMP de 86 Bytes, foi só comprimido

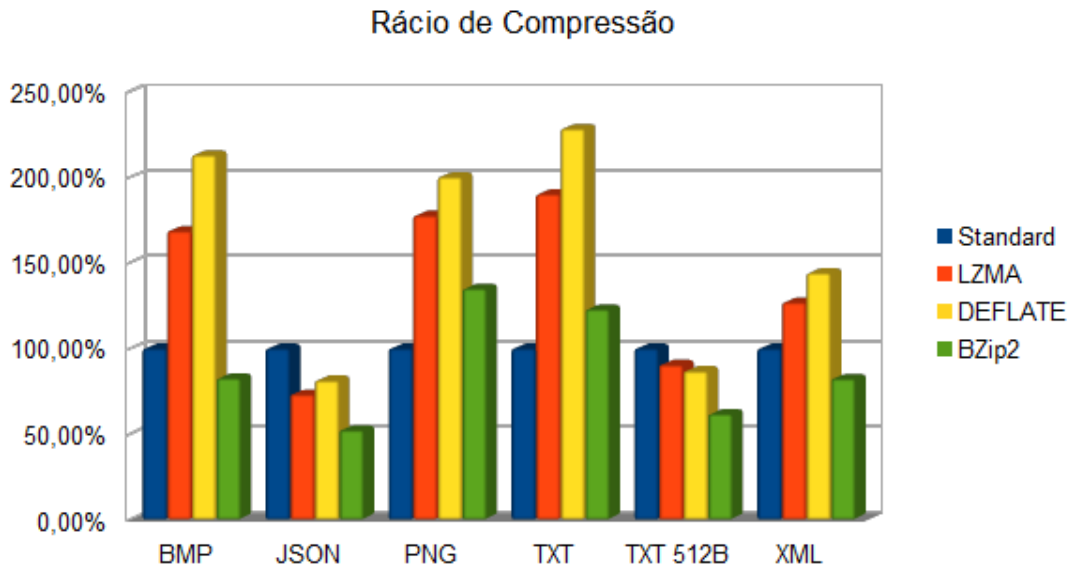

Figura 4.6: Rácio de Compressão por Tipo de Ficheiro (Tabela A.14)

com uma taxa vantajosa pelo algoritmo BZip2, tendo sido obtido nos restantes algoritmos taxas de compressão semelhantes às obtidas com os ficheiros PNG e TXT.

Analisando agora os tempos de cifragem das diferentes cifras, na figura 4.7, é possível verificar que a cifra RSA é extremamente ineficiente quando comparada com as restantes cifras, sendo até quase 4000% mais lenta, quando comparada com a cifra mais rápida, tendo a cifra mais rápida executado num tempo de 0,0017337 milisegundos e a RSA executado em 0,068929 milisegundos.

Analisando as restantes cifras, podemos verificar na figura 4.8, podemos verificar no eixo horizontal que as duas cifras que se destacam como sendo as mais rápidas são a AES e DES.

A cifra Blowfish é a segunda cifra mais lenta, demorando cerca de quatro vezes mais que as duas cifras mais rápidas e duas vezes mais que a DESede.

A cifra DESede demora praticamente o dobro a encriptar quando comparada com as cifras mais rápidas.

Relativamente à decifragem, a cifra RSA continuou com os piores resultados sendo, neste caso, cerca de 200.000% mais lenta que a cifra mais rápida. Por esse motivo, os seus valores não foram contemplados no gráfico 4.9. Devido a estes tempos de execução da cifra RSA, esta cifra não foi incluída no protótipo final, pois os tempos de execução são demasiado elevados.

De forma semelhante ao processo de cifragem, é possível verificar que as duas cifras que se apresentam mais rápidas são a AES e a DES.

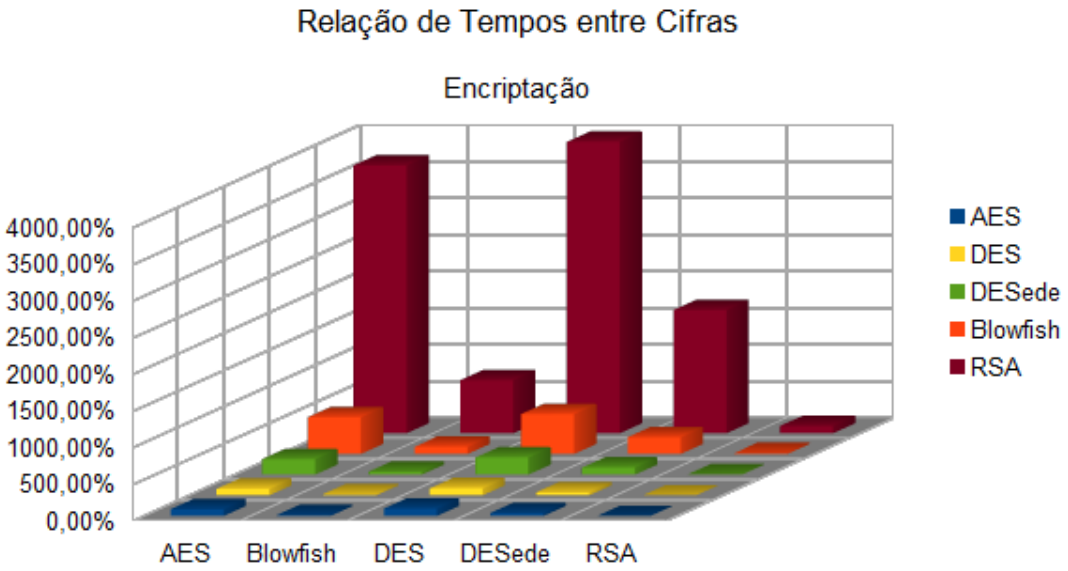

Figura 4.7: Relação de Tempos de cifragem entre Cifras (Tabela A.7)

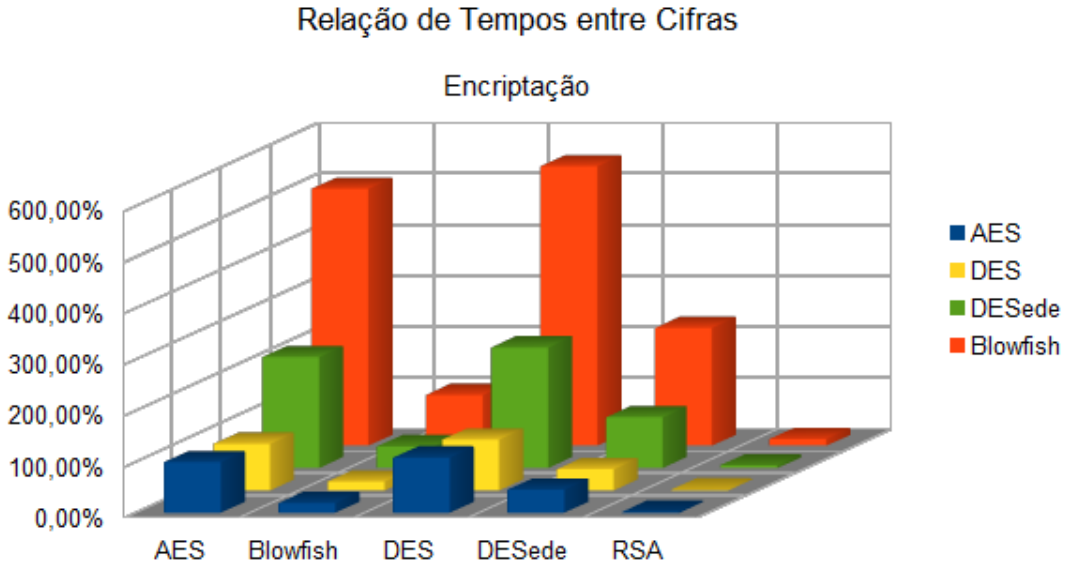

Figura 4.8: Relação de Tempos de cifragem entre Cifras sem RSA(Tabela A.7)

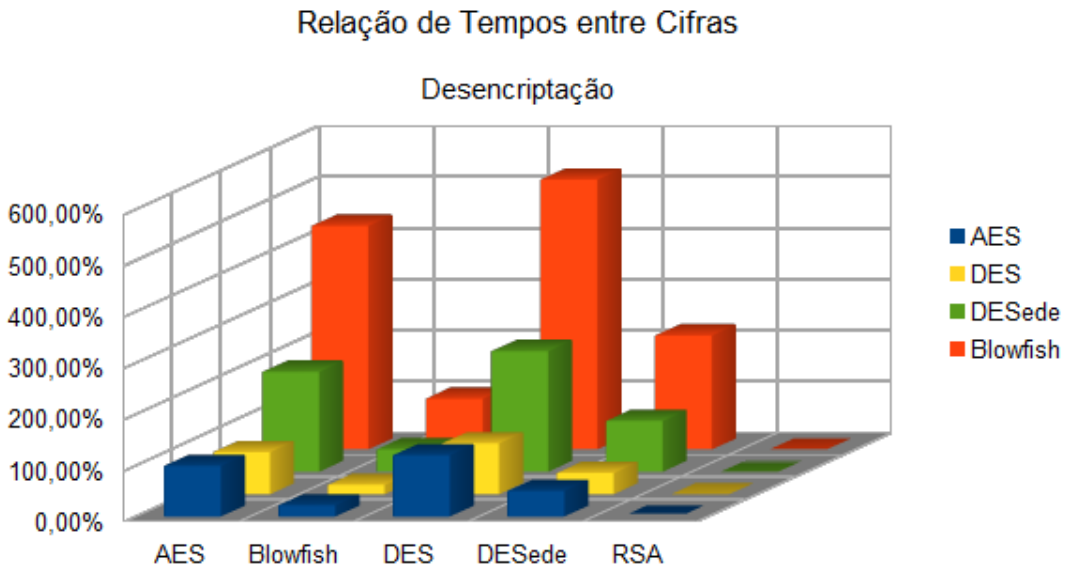

Figura 4.9: Relação de Tempos de decifragem entre Cifras sem RSA(Tabela A.8)

A cifra Blowfish continua a ser das mais lentas, demorando quatro vezes mais que a DES e duas vezes mais que a DESede.

Por fim, comparando o tempo de cifragem com o tempo de decifragem, é possível verificar no gráfico da figura 4.10 que ambos são bastante semelhantes, com uma diferença muito pequena.

A cifra RSA não foi contemplada neste gráfico devido ao facto do tempo necessário para a sua decifragem ser demasiado grande. O gráfico com a cifra poderá ser visto na figura A.2.

Foram também posteriormente também efetuados outros testes de modo a serem medido os tempos de execução das cifras quando combinadas com compressão/descompressão de dados. Estes testes foram efetuados após a recolha dos dados anteriores e, tendo em consideração a performance obtida com a cifra RSA, decidiu-se não incluir esta cifra nestes dados.

Analisando o processo de cifragem com e sem compressão, pode-se verificar pelo gráfico da figura 4.11 que, caso seja efetuada a compressão dos dados, o tempo de execução torna-se substancialmente maior.

Analisando agora o processo de decifragem e descompressão dos dados, presente no gráfico da figura 4.12, é possível ver que, ao contrário do processo de compressão e cifragem, a diferença é bastante pequena, não revelando qualquer impacto significativo.

Tendo em consideração as diferenças obtidas quando comparado o tempo de execução das cifras com e sem compressão/descompressão, pode-se considerar que a diferença não é relevante, principalmente quando se tem em conta o ganho que se pode obter através da

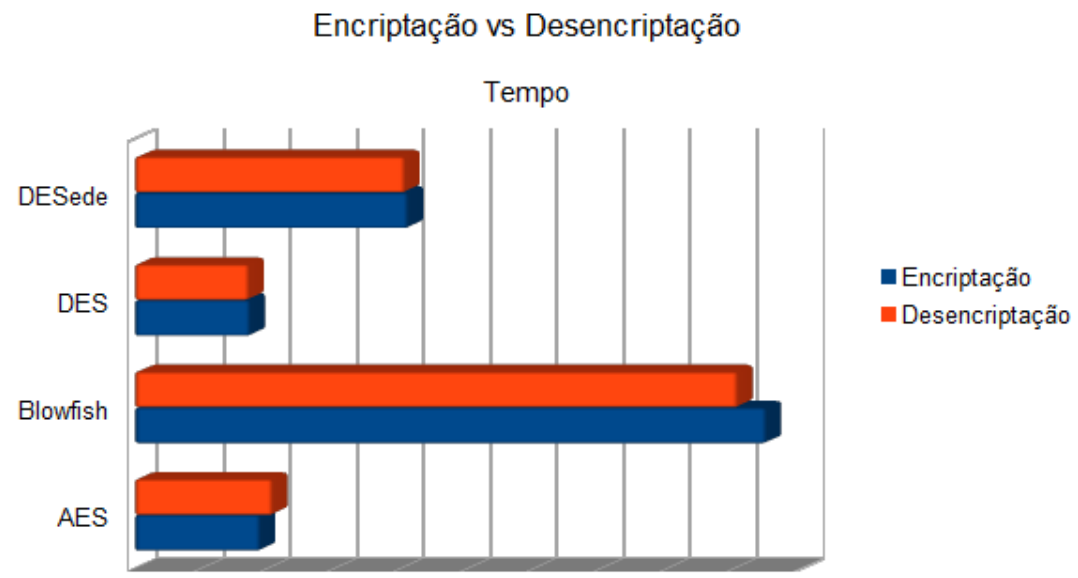

Figura 4.10: cifragem vs decifragem (Tempo)

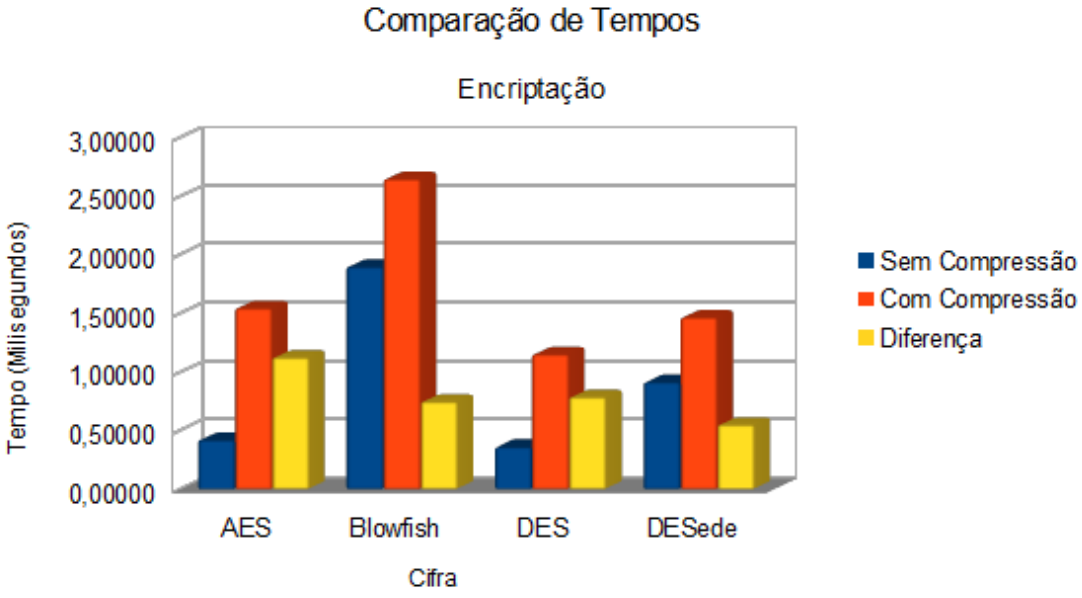

Figura 4.11: Comparação de Tempos de cifragem e Compressão(Milisegundos)

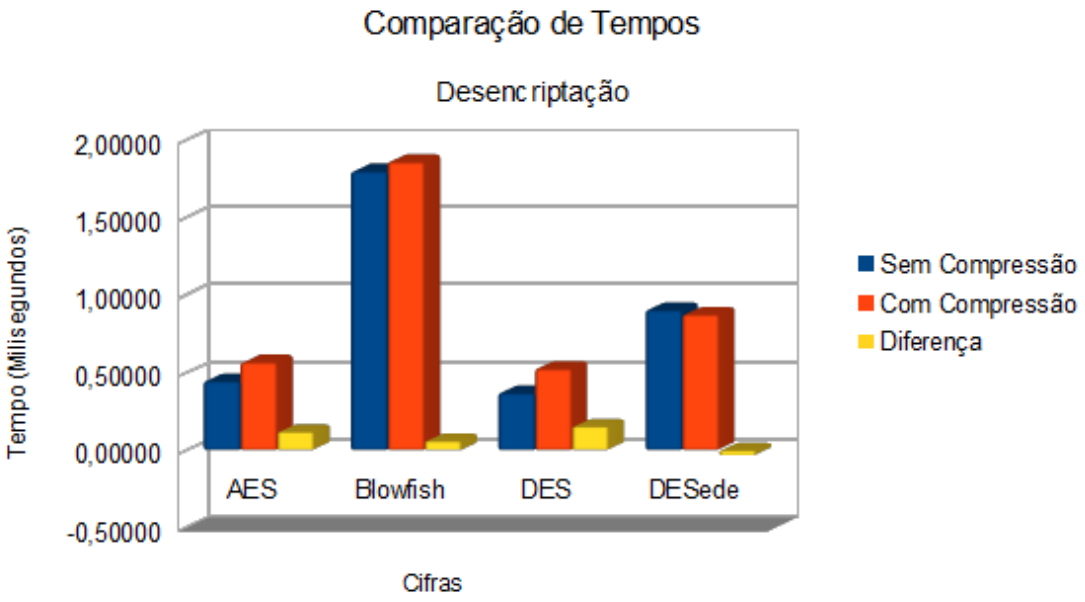

Figura 4.12: Comparação de Tempos de cifragem e Compressão(Milisegundos)

compressão da informação.

Adicionalmente, numa situação de utilização normal, o número de vezes que uma etiqueta NFC será escrita é extremamente reduzido quando comparado com a quantidade de vezes que a etiqueta será lida. Em muitos casos a etiqueta será escrita unicamente uma vez. Por esse motivo, o acréscimo de tempo que se obtém no processo de cifragem mostra-se irrelevante quando comparado com o ganho obtido em termos de espaço. No caso da descompressão que será efetuada no processo de leitura, a diferença é praticamente nula, não tendo impacto visível na performance.

4.4 Conclusões

De acordo com os dados analisados, e pela natureza das operações, é possível concluir que o tipo de compressão que é aplicado a um determinado ficheiro antes da sua cifragem, não influencia de forma direta no desempenho do processamento da cifra aplicada. Isto deve-se principalmente porque os processos de cifragem e decifragem funcionam com base em blocos de dados binários.

Da mesma forma, também se verificou que o tipo de combinação entre o algoritmo de compressão e a cifra de cifragem não tem qualquer influência no resultado final. Desta forma, pode-se concluir que o algoritmo de compressão utilizado irá influenciar unicamente o tamanho do ficheiro a ser encriptado, no entanto, o resultado após a cifragem não tem qualquer relação direta com o algoritmo de compressão utilizado.

Aquilo que a compressão influencia será no tamanho dos dados a serem guardados, o que iria influenciar no tipo de etiqueta NFC a ser usada. Nos casos observados, o único método de compressão que se mostra eficiente na maioria dos casos é o algoritmo BZip2. Contudo, a utilização deste tipo de compressão não garante sempre que o resultado obtido será mais pequeno que o ficheiro original.

Quando analisadas as taxas de compressão por tipo de ficheiro, é possível verificar que o maior número de compressões vantajosas ocorreram com ficheiros de texto com bastante informação. Desta forma, caso o ficheiro a ser armazenado seja informação de texto, quanto maior a quantidade de informação a ser comprimida, maior a probabilidade de se obter compressões mais vantajosas.

Relativamente aos tempos de cifragem e decifragem é possível concluir que a cifra RSA, apesar de ser uma das mais seguras, é a menos eficiente em termos de tempo, demorando extremamente mais a executar as operações que as restantes cifras, principalmente durante o processo de decifragem. Isto levanta um problema devido à sua utilização pois, o processo de decifragem será o processo mais usado, visto que será o processo necessário para a leitura da informação. Sendo este processo bastante lento, afeta a forma como a leitura é efetuada.

Relativamente às restantes cifras, é possível verificar que as duas mais eficientes são o AES e o DES, sendo o DES ligeiramente mais rápido que o AES. Contudo, devido ao facto da cifra DES já ter sido considerada obsoleta e visto a diferença de tempo ser insignificante, recomenda-se que a cifra que seja utilizada para se implementar um processo destes seja a AES. Nenhuma das outras cifras se revela melhor em qualquer outro aspecto.

A utilização da cifra RSA só será vantajosa caso a informação a proteger seja de tal maneira sensível que justifique a utilização de chaves pública e privada e não sendo o tempo um fator importante.

Capítulo 5

Conclusão e Trabalho Futuro

Os processos de cifragem e de compressão são processos complexos que exigem um processamento considerável do dispositivo que os executa.

Foram analisados neste trabalho diferentes algoritmos de cifragem e compressão tanto em termos do tamanho do seu resultado como também em termos do tempo da sua execução. Os resultados obtidos foram positivos, mostrando que é possível criar uma aplicação que utilize compressão e cifragem em ambientes de espaço reduzido, utilizando dispositivos móveis.

De acordo com os dados obtidos é possível determinar que é possível utilizar uma combinação de algoritmos de compressão e cifragem de modo a que se consiga aproveitar ao máximo o espaço limitado das etiquetas NFC, garantindo também a segurança da informação através da utilização de cifras eficientes nos atuais dispositivos móveis.

O caso obtido nos testes que é considerado o mais vantajoso em termos de compressão é a compressão de um ficheiro JSON com o algoritmo BZip2. Nesta situação foi obtido um ficheiro comprimido com 52% do ficheiro original. Esta diferença poderá ser considerável, visto permitir que sejam utilizadas etiquetas NFC de menor capacidade. No entanto, cada caso deverá ser analisado em separado, visto que a natureza dos dados influencia o resultado da compressão.

Quando nos debruçamos sobre os processos de cifragem, verificamos que a maior parte das cifras são executadas em questões de milisegundos, não sendo percetível para o utilizador. A única cifra que claramente se destaca em termos de tempo de execução é a RSA, chegando a demorar dois a três segundos no processo de decifragem. Este tempo poderá ser considerado pouco para muitas situações, mas visto que o tempo é já percetível e que o ganho de segurança não parece justificável para a situação em questão, este algoritmo poderá ser colocado de parte.

Para além do mais, o resultado da cifragem pela cifra RSA gerou sempre ficheiros com um tamanho considerável de 512 Bytes, obrigando desta forma que fossem utilizadas etiquetas NFC de maior capacidade e invalidando qualquer tipo de vantagem em se executar o processo de compressão.

Ainda mais, o nível de complexidade de implementação e gestão desta cifra é extremamente mais complicada, quando comparada com as restantes cifras.

As duas cifras identificadas como as mais eficientes em termos de tempo foram a AES e DES. Estas duas cifras produzem resultados muito semelhantes quer em termos de tempo de execução como no tamanho do ficheiro encriptado, sendo a DES marginalmente melhor nas duas variáveis.

Desta forma, qualquer uma destas cifras seria ideal para implementar no sistema. Contudo, a cifra DES já foi descontinuada e é atualmente considerada insegura por ser possível efetuar ataques de força bruta sem grande dificuldade. Por esse motivo é recomendado que seja utilizada a cifra AES que, para além de ser considerado o melhor método de cifragem de informação em termos de resultado do ficheiro final e tempo de execução, é um dos standards considerados mais seguros da atualidade.

Relativamente à combinação entre o algoritmo de compressão e a cifra de cifragem, não foi encontrada qualquer relação, visto que as cifras funcionam com base em informação binária, não influenciando o tipo de ficheiro processado.

Resumindo, recomendado de acordo com os dados obtidos será, inicialmente aplicar uma compressão BZip2, visto se ter mostrado a melhor e, caso a informação tenha sido comprimida com sucesso e possibilite a utilização de uma etiqueta de menor capacidade, aplica-se a cifragem com a cifra AES. Caso contrário, deverá ser efetuada a cifragem sem qualquer tipo de compressão. Quando visto em termos de tempo de execução, verifica-se que existe uma perda de performance quando é efetuada a compressão e cifragem dos dados. No entanto, visto que este processo só será executado na escrita da etiqueta, que por sua vez será executado uma única vez, a diferença deixa de se mostrar relevante quando comparada com o ganho. Por sua vez, o processo de decifragem e descompressão não tem nenhuma alteração relevante de performance, quando comparada com um processo sem descompressão. Visto também que o processo de leitura será executado muitas mais vezes e não havendo diferença relevante nos tempos de execução, mostra-se vantajoso a utilização de algoritmos de compressão com cifras.

5.1 Satisfação dos Objetivos

Pretendia-se com este trabalho, determinar uma combinação, eficiente tanto a nível de tamanho como a nível de tempo de processamento. Após a recolha e análise dos dados, determinou-se que, relativamente ao tamanho, o algoritmo que se mostra mais eficiente é

o BZip2, enquanto que, relativamente à cifragem, o que se apresenta mais equilibrado em termos de tempo, segurança e tamanho do ficheiro resultante é a cifra AES.

Desta forma, é seguro afirmar que os objetivos deste trabalho foram cumpridos. Serve como contributo as análises efetuadas e os resultados obtidos com este trabalho, permitindo a quem queira implementar diferentes algoritmos de compressão e cifragem em dispositivos móveis, tenha alguma informação para se basear.

Contudo, ainda existe muitas outras formas de se analisar este problema e outros algoritmos, tanto de compressão como de cifragem, que podem ser explorados, sendo ainda possível desenvolver mais este trabalho.

5.2 Trabalho Futuro

Como sugestão para trabalho futuro, sugere-se que sejam analisados outras cifras menos conhecidas, assim como outros diferentes algoritmos de compressão que se possam mostrar mais eficientes em dispositivos móveis.

Poderá também ser analisado, para além da eficiência do tamanho do ficheiro resultante da compressão, o tempo que o processo de compressão e descompressão necessita e ter essa variante também em consideração.

Outra possibilidade para um trabalho futuro, seria o estudo e possível implementação de diferente níveis de permissões de leitura nas etiquetas NFC, permitindo criar múltiplas chaves que iriam revelar diferentes conjuntos de informação encriptada, guardada na etiqueta NFC.

Referências

[ANR74] N. Ahmed, T. Natarajan e K.R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, C-23(1):90–93, January 1974.

[BCW90] Timothy C. Bell, John G. Cleary e Ian H. Witten. Text compression. January 1990.

[Bir04] A Biryukov. *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[BKY02] Enrico Buonanno, Jonathan Katz e Moti Yung. Incremental Unforgeable Encryption. *Proceedings of the 8th International Workshop on Fast Software Encryption*, 2355:109–124, 2002.

[Bon99] Dan Boneh. Twenty Years of attacks on the RSA Cryptosystem. *Notices of the American Mathematical Society*, 46(2):203 – 213, 1999.

[BPW10] Achim D Brucker, Helmut Petritsch e Stefan G Weber. *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, volume 6033 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010.

[Bru93] Bruce Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). *Fast Software Encryption, Cambridge Security Workshop Proceedings*, pages 191 – 204, 1993.

[BS91] Eli Biham e Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[BW94] M. Burrows e D. J. Wheeler. A Block-sorting Lossless Data Compression Algorithm. Technical report, Systems Research Center, California, 1994.

[BWC89a] Timothy Bell, Ian H. Witten e John G. Cleary. Modeling for text compression. *ACM Computing Surveys*, 21(4):557–591, December 1989.

[BWC89b] Timothy Bell, Ian H. Witten e John G Cleary. Modeling for Text Compression. 21(4), 1989.

[CBSU06] R. Chandramouli, S. Bapatla, K. P. Subbalakshmi e R. N. Uma. Battery power-aware encryption. *ACM Transactions on Information and System Security*, 9(2):162–180, May 2006.

[CIHC06] Guoqin Cui Guoqin Cui, A M Ivanovic, T S Huang e L Chen. Using data compression to enhance the security of identification document, 2006.

[CN07] Marton Csapodi e Andras Nagy. New Applications for NFC Devices. *2007 16th IST Mobile and Wireless Communications Summit*, pages 1–5, 2007.

[Col05] Lasse Collin. A Quick Benchmark: Gzip vs. Bzip2 vs. LZMA. *[http://tukaani.org/ The Tukaani Project]*, 2005.

[Dah07] McConnachie Dahna. Bruce Almighty: Schneier preaches security to Linux faithful. *Computerworld*, 2007.

[DH77] W. Diffie e M.E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, June 1977.

[DKK+11] Sebastian Dunnebeil, Felix Kobler, Philip Koene, Jan Marco Leimeister e Helmut Krcmar. Encrypted NFC Emergency Tags Based on the German Telematics Infrastructure, 2011.

[DRC+10] Kenny Daily, Paul Rigor, Scott Christley, Xiaohui Xie e Pierre Baldi. Data structures and compression algorithms for high-throughput sequencing technologies. *BMC bioinformatics*, 11(1):514, January 2010.

[DW02] I Devetak e A Winter. Classical data compression with quantum side information. *Physical Review A*, 68(4):10, 2002.

[eaZ67] et. al. Zwicker. *The Ear As A Communication Receiver*. Acoustical Society of America, Melville, NY, 1967.

[EYCP01] A.J. Elbirt, W. Yip, B. Chetwynd e C. Paar. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(4):545–557, August 2001.

[Fin94] C Finnila. Combining data compression and encryption, 1994.

[Fli06] David Flint. RFID tags, security and the individual. *Computer Law Security Report*, 22(2):165–168, 2006.

[Gir09] Bernd Girod. lossless compression. *Image Rochester NY*, 169(2):1–10, 2009.

[Han08] Gerhard P Hancke. *4th Workshop on RFID Security (RFIDsec'08)*. 2008.

[HANM10] M A S Hernandez, O Alvarado-Nava e F J Z Martinez. Huffman Coding-Based Compression Unit for Embedded Systems, 2010.

[HL88] D R Helman e G G Jr Langdon. Data compression, 1988.

[HL11] Martin Hilbert e Priscila López. The world's technological capacity to store, communicate, and compute information. *Science (New York, N.Y.)*, 332(6025):60–5, April 2011.

[HSC07] M A Haleem, K P Subbalakshmi e Rajarathnam Chandramouli. Joint Encryption and Compression of Correlated Sources with Side Information. *EURASIP Journal on Information Security*, 2007, 2007.

[IRP08] M R Islam, S A Ahsan Rajon e A Podder. Short text compression for smart devices, 2008.

[Iso08] Minna Isomursu. Tags and the City. *PsychNology Journal*, 6(2):131–156, 2008.

[ISO10] ISO. *ISO/IEC 18033-3:2010 Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*. Iso.org, 2010.

[JAZS10] Antonio J Jara, Alberto F Alcolea, Miguel A Zamora e Antonio F G Skarmeta. Evaluation of the security capabilities on NFC-powered devices. *Smart Objects Systems Technologies and Applications RFID Sys Tech 2010 European Workshop on*, pages 1–9, 2010.

[JG08] Y K Jain e P B Gosavi. Email Security Using Encrption and Compression, 2008.

[JGHQ11] Jiang Jian, Liao Guoqiong, Wang Hao e Wan Qizhi. A RFID Mixed Coding Scheme Based on Huffman Algorithm, 2011.

[Joh03] Thomas Johansson, editor. *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[Joh06] Don Johnson. Compression and the Huffman Code. *Source*, (1):91–93, 2006.

[JP09] Amit Jain e Ravindra Patel. An Efficient Compression Algorithm (ECA) for Text Data. In *2009 International Conference on Signal Processing Systems*, pages 762–765. IEEE, 2009.

[KJKB09] K Koscher, A Juels, T Kohno e V Brajkovic. EPC RFID Tags in Security Applications : Passport Cards , Enhanced Drivers Licenses , and Beyond. *ACM Conference on Computer and Communications Security*, pages 33–42, 2009.

[KM97] H. Kruse e A. Mukherjee. Data compression using text encryption. In *Proceedings DCC '97. Data Compression Conference*, page 447. IEEE Comput. Soc. Press, 1997.

[LCCL05] Shujun Li, Guanrong Chen, Albert Cheung e Kwok-Tung Lo. Cryptanalysis of an MPEG-Video Encryption Scheme Based on Secret Huffman Tables. *Computer*, 5414:8, 2005.

[LD08] David Lisonek e Martin Drahanský. SMS Encryption for Mobile Communication. In *2008 International Conference on Security Technology*, pages 198–201. IEEE, December 2008.

[Mat94] Mitsuru Matsui. *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, July 1994.

[MBP+09] J Muller, V Borovskiy, O Panchenko, A Bog e A Zeier. NFC at the workplace amp;#x2014; Simplify enterprise work flows with NFC, 2009.

[Men96] Alfred Menezes. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MH81] Ralph Merkle e Martin Hellman. On the Security of Multiple Encryption. *Communications of the ACM*, 24(7):465 – 467, 1981.

[MM94] Mitsuru Matsui e Mitsuru Matsui. *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, July 1994.

[MM12] O.A. Mahdi e O.A Mahdi. Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique. *International Journal of Computer Science Issues*, 9(6, No. 3):53 – 59, 2012.

[NJ05] A. Nadeem e M.Y. Javed. A Performance Comparison of Data Encryption Algorithms. In *2005 International Conference on Information and Communication Technologies*, pages 84–89. IEEE, 2005.

[Nos11] Dan Nosowitz. Everything You Need to Know About Near Field Communication. *Popular Science Magazine*, 2011.

[Ort06] C. Enrique Ortiz. *An Introduction to Near-Field Communication and the Contactless Communication API*. 2006.

[PAP06] This Postnote, Investigatory Powers Act e Internet Protocol. Data encryption. *Policy*, (270), 2006.

[PRA+12] Mikko Pyykkönen, Jukka Riekki, Ismo Alakärppä, Ivan Sanchez, Marta Cortes e Sonja Saukkonen. Designing Tangible User Interfaces for NFC Phones. *Advances in HumanComputer Interaction*, 2012:1–12, 2012.

[Riv95] Ronald L Rivest. The RC5 Encryption Algorithm. *Technology*, 1008(1008):86–96, 1995.

[Riv97] Ronald L Rivest. 2 A Parameterized Family of Encryption Algorithms. 1997.

[ROSC08] H Rodriguez, B Ontiveros, I Soto e R Carrasco. A public key encryption model for wireless sensor networks. *Communication Systems Networks and Digital Signal Processing 2008 CNSDSP 2008 6th International Symposium on*, pages 373–377, 2008.

[RSA78] R. L. Rivest, A. Shamir e L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[S. 06] J. Pelzl G. Pfeiffer A. Rupp M. Schimmler S. Kumar, C. Paar. How to Break DES for Euro 8,980 - 2nd Workshop on Special-purpose Hardware for Attacking Cryptographic Systems, 2006.

[Say00] Khalid Sayood. *Introduction to Data Compression*. The Morgan Kaufmann Series in Multimedia Information and Systems. Morgan Kaufmann Publishers, 2000.

[Say12] Michael Saylor. *The Mobile Wave: How Mobile Intelligence Will Change Everything*. Perseus Books/Vanguard Press, 2012.

[Sol06] David Solomon. *Data Compression: The Complete Reference*. Springer, 2006.

[Spe06] Technical Specification. NFC Data Exchange Format (NDEF) Technical Specification. 2006.

[Spe11] Technical Specification. Simple NDEF Exchange Protocol Technical Specification. 2011.

[TG07] Tom Gonzalez e Tom Gonzalez. *A Reection Attack on Blowsh*. JOURNAL OF LATEX CLASS FILES, 2007.

[TJ09] Thomas R. Johnson e Thomas R Johnson. *American Cryptology during the Cold War, 1945-1989.Book III: Retrenchment and Reform, 1972-1980, page 232*. NSA, DOCID 3417193 (file released on 2009-12-18, hosted at cryptome.org), 2009.

[TPP06] G Tim, Christof Paar e Jan Pelzl. On the security of elliptic curve cryptosystems against attacks with special-purpose hardware. *2nd Workshop on Specialpurpose Hardware for Attacking Cryptographic Systems SHARCS 2006*, pages 1–20, 2006.

[Ver08] Roel Verdult. Security analysis of RFID tags. *Master Thesis*, page 50, 2008.

[VR97] Vincent Rijmen e Vincent Rijmen. Cryptanalysis and Design of Iterated Block Ciphers. *Ph.D thesis*, 1997.

[VV12] Jessica E. Vascellaro e Jessica E Vascellaro. Inside Apple's Go-Slow Approach to Mobile Payments. *The Wall Street Journal*, 2012.

[Wad94] Graham Wade. *Signal coding and processing*. Cambridge University Press, 1994.

[WT97a] Walter Tuchman e Walter Tuchman. *Internet besieged: countering cyberspace scofflaws*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997.

[WT97b] Walter Tuchman e Walter Tuchman. *Internet besieged: countering cyberspace scofflaws*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1997.

[WY10] Suli Wu Suli Wu e Xiaofei Yi Xiaofei Yi. Text Encryption Algorithm Based Cyclic Shift, 2010.

[YYJY11] Mo Yuanbin, Qiu Yubing, Liu Jizhong e Ling Yanxia. A Data Compression Algorithm Based on Adaptive Huffman Code for Wireless Sensor Networks, 2011.

[ZZH09] Naidong Zhao, Runtong Zhang e Ling Han. A Combination of Encryption and Compression Algorithm Based on Huffman. In FG Duserick, editor, *EIGHTH WUHAN INTERNATIONAL CONFERENCE ON EBUSINESS VOLS IIII*, pages 411–415. China Univ Geosci; CICEB; Coll Econ & Management; Coll Business; Alfred Univ, ALFRED UNIV, 2009.

Anexo A

Tabelas e Resultados Obtidos

Tipo	Compressão	Standard	AES	Blowfish	DES	DESede	RSA
BMP	N/A	86	96	88	88	88	512
	LZMA	145	160	152	152	152	512
	DEFLATE	183	192	184	184	184	512
	BZip2	71	80	72	72	72	512
JSON	N/A	432	448	440	440	440	512
	LZMA	316	320	320	320	320	512
	DEFLATE	351	352	352	352	352	512
	BZip2	226	240	232	232	232	512
PNG	N/A	138	144	144	144	144	512
	LZMA	245	256	248	248	248	512
	DEFLATE	276	288	280	280	280	512
	BZip2	186	192	192	192	192	512
TXT Small	N/A	110	112	112	112	112	512
	LZMA	209	224	216	216	216	512
	DEFLATE	251	256	256	256	256	512
	BZip2	135	144	136	136	136	512
TXT 512B	N/A	512	528	520	520	520	512
	LZMA	463	464	464	464	464	512
	DEFLATE	444	448	448	448	448	512
	BZip2	316	320	320	320	320	512
XML	N/A	198	208	200	200	200	512
	LZMA	251	256	256	256	256	512
	DEFLATE	285	288	288	288	288	512
	BZip2	163	176	168	168	168	512

Tabela A.1: Tamanhos obtidos (Unidade: Bytes)

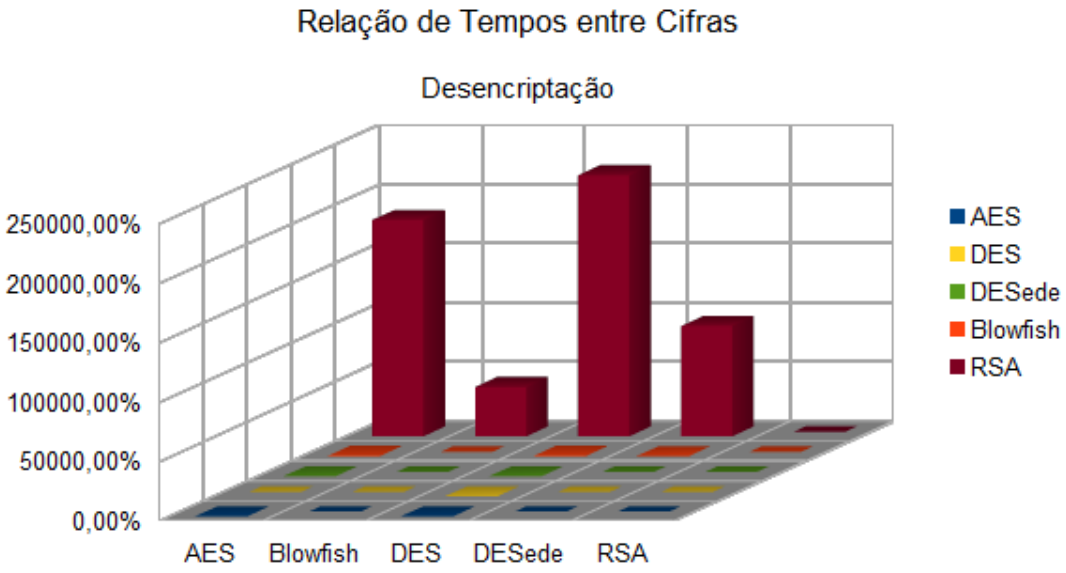

Figura A.1: Relação de Tempos de Desencriptação entre Cifras (Tabela A.8)

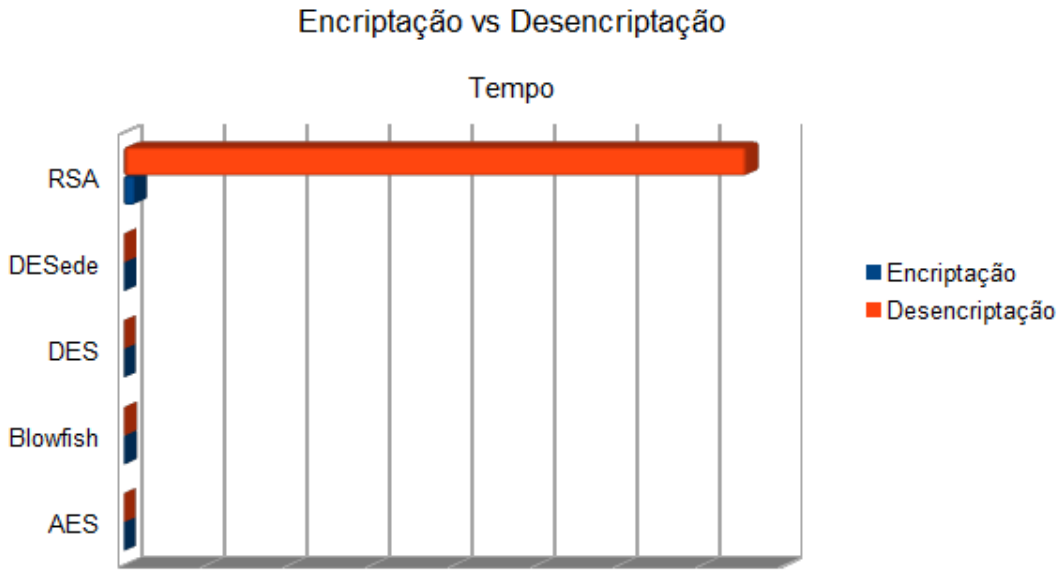

Figura A.2: Encriptação vs Desencriptação com RSA (Tempo)

		Standard	AES	Blowfish	DES	DESede	RSA
BMP	Standard	0,000	0,322	1,437	0,210	0,520	20,000
	LZMA	0,000	0,370	2,224	0,332	0,590	18,000
	DEFLATE	0,000	0,325	2,025	0,326	0,906	49,000
	BZip2	0,000	0,262	1,983	0,348	0,607	131,000
JSON	Standard	0,000	0,512	2,417	0,559	1,255	12,000
	LZMA	0,000	0,492	1,547	0,382	1,189	17,000
	DEFLATE	0,000	0,594	1,864	0,334	0,873	18,000
	BZip2	0,000	0,400	1,738	0,387	0,998	75,000
PNG	Standard	0,000	0,418	1,940	0,258	0,543	9,000
	LZMA	0,000	0,575	2,220	0,374	0,923	7,000
	DEFLATE	0,000	0,427	1,660	0,317	0,871	19,000
	BZip2	0,000	0,270	1,398	0,265	0,609	34,000
TXT	Standard	0,000	0,374	1,746	0,359	0,755	42,000
	LZMA	0,000	0,307	1,516	0,289	0,714	24,000
	DEFLATE	0,000	0,416	2,045	0,362	0,925	17,000
	BZip2	0,000	0,376	2,068	0,372	0,634	36,000
TXT 512B	Standard	0,000	0,670	2,444	0,519	1,481	18,000
	LZMA	0,000	0,628	2,210	0,449	1,188	17,000
	DEFLATE	0,000	0,406	1,549	0,362	1,038	132,000
	BZip2	0,000	0,469	1,585	0,462	1,350	15,000
XML	Standard	0,000	0,375	1,930	0,372	0,904	50,000
	LZMA	0,000	0,436	2,066	0,361	1,159	16,000
	DEFLATE	0,000	0,436	2,356	0,321	0,978	29,000
	BZip2	0,000	0,275	1,619	0,376	0,922	42,000

Tabela A.2: Tempos da Encriptação (Milisegundos)

		Standard	AES	Blowfish	DES	DESede	RSA
BMP	Standard	0,000	0,395	1,976	0,282	0,599	1758,000
	LZMA	0,000	0,330	1,477	0,324	0,592	2577,000
	DEFLATE	0,000	0,439	1,768	0,362	0,828	2401,000
	BZip2	0,000	0,408	1,869	0,269	0,602	2684,000
JSON	Standard	0,000	0,606	2,222	0,510	1,334	1308,000
	LZMA	0,000	0,456	2,127	0,577	1,153	1797,000
	DEFLATE	0,000	0,398	1,772	0,517	1,168	2060,000
	BZip2	0,000	0,549	1,858	0,304	0,955	1782,000
PNG	Standard	0,000	0,300	1,697	0,244	0,784	1162,000
	LZMA	0,000	0,407	1,549	0,326	0,852	1760,000
	DEFLATE	0,000	0,431	1,628	0,374	0,863	2518,000
	BZip2	0,000	0,317	1,426	0,270	0,637	1817,000
TXT	Standard	0,000	0,336	1,581	0,305	0,516	2530,000
	LZMA	0,000	0,382	1,591	0,312	0,795	1534,000
	DEFLATE	0,000	0,365	1,974	0,330	0,780	1556,000
	BZip2	0,000	0,372	1,979	0,252	0,583	1906,000
TXT 512B	Standard	0,000	0,550	2,289	0,560	1,643	1279,000
	LZMA	0,000	0,560	1,699	0,425	1,120	2592,000
	DEFLATE	0,000	0,456	1,587	0,403	1,106	1713,000
	BZip2	0,000	0,518	2,081	0,430	1,033	2046,000
XML	Standard	0,000	0,493	1,550	0,366	1,007	2438,000
	LZMA	0,000	0,597	1,813	0,341	0,832	1301,000
	DEFLATE	0,000	0,608	2,273	0,424	1,118	2126,000
	BZip2	0,000	0,469	1,484	0,411	0,900	1713,000

Tabela A.3: Tempos da Desencriptação (Milisegundos)

	Standard	AES	Blowfish	DES	DESede	RSA
Standard	246,00	256,00	250,67	250,67	250,67	512,00
LZMA	271,50	280,00	276,00	276,00	276,00	512,00
DEFLATE	298,33	304,00	301,33	301,33	301,33	512,00
BZip2	182,83	192,00	186,67	186,67	186,67	512,00

Tabela A.4: Análise de Compressões por Cifras

	Standard	AES	Blowfish	DES	DESede	RSA
Média	0,0000000	0,0018848	0,0094783	0,0017337	0,0041030	0,0689290

Tabela A.5: Tempo Médio de Encriptação por Byte (Milisegundos)

Standard	AES	Blowfish	DES	DESede	RSA
0,0000000	0,0020695	0,0090480	0,0017153	0,0040552	3,7726237

Tabela A.6: Tempo Médio de Desencriptação por Byte (Milisegundos)

	AES	Blowfish	DES	DESede	RSA
AES	100,00%	19,89%	108,72%	45,94%	2,73%
Blowfish	502,88%	100,00%	546,71%	231,01%	13,75%
DES	91,98%	18,29%	100,00%	42,25%	2,52%
DESede	217,69%	43,29%	236,67%	100,00%	5,95%
RSA	3657,11%	727,23%	3975,87%	1679,95%	100,00%

Tabela A.7: Relação de Tempo de Encriptação entre Cifras

	AES	Blowfish	DES	DESede	RSA
AES	100,00%	22,87%	120,65%	51,03%	0,05%
Blowfish	437,20%	100,00%	527,48%	223,12%	0,24%
DES	82,89%	18,96%	100,00%	42,30%	0,05%
DESede	195,95%	44,82%	236,41%	100,00%	0,11%
RSA	182294,56%	41695,80%	219936,40%	93032,59%	100,00%

Tabela A.8: Relação de Tempo de Desencriptação entre Cifras

	AES	Blowfish	DES	DESede	RSA	Média
Standard	3	23	4	6	25	12
LZMA	7	12	5	11	17	10
DEFLATE	6	18	6	9	44	16
BZip2	7	18	4	5	56	18
Média	6	18	5	8	35	

Tabela A.9: Tempo de Encriptação por tipo de compressão

	AES	Blowfish	DES	DESede	RSA	Média
BMP	2	5	8	23	2355	478
JSON	4	7	15	18	1737	356
PNG	1	2	5	4	1814	365
TXT Small	3	4	7	5	1882	380
TXT 512B	3	6	5	12	1908	387
XML	5	3	2	4	1895	382
Média	3	4	7	11	1932	

Tabela A.10: Tempo de Desencriptação por tipo de ficheiro

	AES	Blowfish	DES	DESede	RSA	Média
Standard	1	4	3	13	1746	353
LZMA	2	5	6	8	1927	389
DEFLATE	3	4	6	20	2062	419
BZip2	6	4	13	3	1991	403
Média	3	4	7	11	1932	

Tabela A.11: Tempo de Desencriptação por tipo de compressão

Tipo	Compressão	Standard	AES	Blowfish	DES	DESede	RSA
BMP	BZip2	71	8	14	3	4	131
BMP	Standard	86	4	11	1	2	20
TXT Small	Standard	110	2	7	1	3	42
TXT Small	BZip2	135	2	10	1	2	36
PNG	Standard	138	1	9	1	2	9
BMP	LZMA	145	11	7	4	2	18
XML	BZip2	163	9	13	5	15	42
BMP	DEFLATE	183	7	7	5	13	49
PNG	BZip2	186	12	8	1	3	34
XML	Standard	198	5	27	9	18	50
TXT Small	LZMA	209	2	12	4	30	24
JSON	BZip2	226	6	41	6	3	75
PNG	LZMA	245	9	9	5	5	7
TXT Small	DEFLATE	251	6	10	5	3	17
XML	LZMA	251	6	15	1	2	16
PNG	DEFLATE	276	2	11	7	3	19
XML	DEFLATE	285	1	19	8	8	29
JSON	LZMA	316	9	15	8	12	17
TXT 512B	BZip2	316	3	19	8	3	15
JSON	DEFLATE	351	7	12	7	12	18
JSON	Standard	432	2	28	5	6	12
TXT 512B	DEFLATE	444	12	49	2	14	132
TXT 512B	LZMA	463	4	16	7	13	17
TXT 512B	Standard	512	6	54	7	5	18

Tabela A.12: Relação entre tamanho e tempo de encriptação

	AES	Blowfish	DES	DESede	RSA	Média
BMP	8	10	3	5	55	16
JSON	6	24	7	8	31	15
PNG	6	9	4	3	17	8
TXT Small	3	10	3	10	30	11
TXT 512B	6	35	6	9	46	20
XML	5	19	6	11	34	15
Média	6	18	5	8	35	

Tabela A.13: Tempo de Encriptação por tipo de ficheiro

	Standard	LZMA	DEFLATE	BZip2
BMP	86	168,60%	212,79%	82,56%
JSON	432	73,15%	81,25%	52,31%
PNG	138	177,54%	200,00%	134,78%
TXT	110	190,00%	228,18%	122,73%
TXT 512B	512	90,43%	86,72%	61,72%
XML	198	126,77%	143,94%	82,32%

Tabela A.14: Rácio de Compressão por Tipo de Ficheiro

	AES	Blowfish	DES	DESede
BMP	1,58	2,66	1,26	1,57
JSON	1,58	2,66	1,26	1,57
PNG	1,35	2,77	1,06	1,32
TXT Small	1,43	2,51	1,28	1,43
TXT 512B	1,92	2,96	1,12	1,74
XML	1,42	2,33	0,93	1,17

Tabela A.15: Tempos de Execução com Compressão e Encriptação

	AES	Blowfish	DES	DESede
BMP	0,45	1,81	0,40	0,62
JSON	0,68	2,26	0,62	1,01
PNG	0,50	1,66	0,46	0,79
TXT Small	0,52	1,83	0,49	0,78
TXT 512B	0,69	1,89	0,69	1,24
XML	0,58	1,73	0,51	0,83

Tabela A.16: Tempos de Execução com Desencriptação e Descompressão